# SSH Tectia Server (A/F/T) 5.0

# Administrator Manual

**19 September 2005**

# SSH Tectia Server (A/F/T) 5.0: Administrator Manual

19 September 2005

SSH Communications Security Corp.
Valimotie 17, FIN-00380 Helsinki; Finland

# Table of Contents

**1. About This Document** ........ 7
1.1. Component Terminology ........ 8
1.2. Documentation Conventions ........ 10
1.3. Customer Support ........ 10
**2. Installing SSH Tectia Server** ........ 13
2.1. Planning the Installation ........ 13
2.1.1. System Requirements ........ 13
2.1.2. Packaging ........ 14
2.1.3. Licensing ........ 14
2.1.4. Upgrading from Version 4.x to 5.0 ........ 15
2.1.5. Upgrading from Version 5.0 ........ 15
2.2. Installing the SSH Tectia Server Software ........ 16
2.2.1. Installing on AIX ........ 16
2.2.2. Installing on HP-UX ........ 17
2.2.3. Installing on Linux ........ 18
2.2.4. Installing on Solaris ........ 18
2.2.5. Installing on Windows ........ 19
2.3. Removing the SSH Tectia Server Software ........ 20
2.3.1. Removing from AIX ........ 20
2.3.2. Removing from HP-UX ........ 21
2.3.3. Removing from Linux ........ 21
2.3.4. Removing from Solaris ........ 22
2.3.5. Removing from Windows ........ 22
**3. Getting Started** ........ 23
3.1. Location of Installed Files ........ 23
3.1.1. File Locations on Unix ........ 23
3.1.2. File Locations on Windows ........ 24
3.2. Starting and Stopping the Server ........ 24
3.2.1. Starting and Stopping on Unix ........ 25
3.2.2. Starting and Stopping on Windows ........ 25
3.3. Examples of Use ........ 26

**4. Configuring SSH Tectia Server** ..... 27
ssh-server-config ..... 27
4.1. Configuration Tool (Windows) ..... 52
4.1.1. SSH Tectia Server ..... 52
4.1.2. General ..... 54
4.1.3. Identity ..... 56
4.1.4. Network ..... 58
4.1.5. Logging ..... 60
4.1.6. Certificate Validation ..... 61
4.1.7. Editing Selectors ..... 64
4.1.8. Connections and Encryption ..... 67
4.1.9. Authentication ..... 69
4.1.10. Services ..... 73
**5. Authentication** ..... 79
5.1. Server Authentication with Public Keys ..... 79
5.1.1. Generating the Host Key ..... 81
5.1.2. Notifying the Users of the Host Key Change ..... 81
5.2. Server Authentication with Certificates ..... 82
5.2.1. Certificate Enrollment Using `ssh-cmpclient` ..... 83
5.3. Server Authentication using External Host Keys ..... 84
5.4. User Authentication with Passwords ..... 84
5.4.1. Special Considerations on Windows ..... 85
5.5. User Authentication with Public Keys ..... 85
5.5.1. Special Considerations on Windows ..... 86
5.6. User Authentication with Certificates ..... 86
5.6.1. Certificate Configuration ..... 87
5.7. Host-Based User Authentication ..... 89
5.7.1. Using Traditional Public Keys ..... 90
5.7.2. Using Certificates ..... 92
5.8. User Authentication with Keyboard-Interactive ..... 93
5.8.1. Password Submethod ..... 94
5.8.2. Pluggable Authentication Module (PAM) Submethod ..... 94
5.8.3. RSA SecurID Submethod ..... 95
5.8.4. RADIUS Submethod ..... 96
5.9. User Authentication with GSSAPI ..... 96
**6. System Administration** ..... 99
6.1. SSH Tectia Client Privileged User ..... 99
6.1.1. Disabling Root Login ..... 99
6.1.2. Restricting Connections ..... 100
6.1.3. Forced Commands ..... 100
6.2. Auditing ..... 101
6.2.1. Notification ..... 102
6.2.2. Customizing Logging ..... 102

**7. File Transfer** .......... 105
7.1. SSH Tectia Client File Transfer User .......... 106
7.1.1. Encryption and Authentication Methods .......... 106
7.1.2. Restricting Services .......... 107
7.1.3. Settings on the Client Side .......... 108
7.2. Automated File Transfer .......... 109
**8. Tunneling** .......... 111
8.1. SSH Tectia Connector Tunneling User .......... 111
8.1.1. Using a Shared Account .......... 112
8.1.2. Restricting Services .......... 112
8.2. Local Tunnels .......... 114
8.3. Remote Tunnels .......... 116
8.4. X11 Forwarding (Unix) .......... 117
8.5. Agent Forwarding (Unix) .......... 118
A. Command-Line Tools .......... 119
ssh-server-g3 .......... 119
ssh-server-config-tool .......... 121
ssh-keygen-g3 .......... 122
ssh-certview-g3 .......... 125
ssh-cmpclient-g3 .......... 128
ssh-ekview-g3 .......... 135
B. Server Configuration File Syntax .......... 137
C. Man Pages and Help Files .......... 149
D. Audit Messages .......... 151
Index .......... 181

# Chapter 1 About This Document

SSH Tectia Server (formerly SSH Secure Shell) is a server-side component for SSH Tectia Connector and Client. There are four separate versions of the product available:

- SSH Tectia Server (A) designed especially for system administration
- SSH Tectia Server (F) designed especially for file transfer
- SSH Tectia Server (T) designed especially for application tunneling
- SSH Tectia Server (M) for IBM mainframes

Table 1.1 shows the main features of the different SSH Tectia Server versions.

**Table 1.1. SSH Tectia Server versions and features**

| **Functionality** | **Server (A)** | **Server (F)** | **Server (T)** | **Server (M)** |
|---|---|---|---|---|
| System administration functionality * | x | x | x | x |
| Client-side tools and functionality | | x | x | x |
| SFTP API | | x | x | x |
| Checkpoint/restart mechanism | | x | x | x |
| Streaming for high-speed transfers | | x | x | x |
| CryptiCore encryption (Intel only) | | x | x | |
| Transparent tunneling (requires SSH Tectia Connector) | | | x | x |
| Support for centralized management (requires SSH Tectia Manager) | x | x | x | |
| **Supported platforms** | | | | |
| Windows | x | x | x | |
| Unix, Linux | x | x | x | |
| IBM Linux on POWER and zSeries | | | x | |
| IBM z/OS | | | | x |

* Includes secure terminal, SFTP server, and (non-transparent) tunneling server.

This document contains instructions on the basic administrative tasks of SSH Tectia Server (A), Server (F), and Server (T). The document is intended for system administrators responsible for the configuration of the SSH Tectia Server software.

To fully use the information presented in this document, you should be familiar also with other system administration tasks. To edit the configuration files manually without SSH Tectia Server Windows GUI or SSH Tectia Manager, you should have basic knowledge of XML.

This document contains the following information:

- Installing SSH Tectia Server
- Getting started
- Configuring SSH Tectia Server
- Authentication settings
- System administration
- File transfer
- Application tunneling
- Troubleshooting
- Appendices, including command-line tool and audit message references

*SSH Tectia Client/Server Product Description* contains important background information on the SSH Tectia client/server solution, and we recommend that you read it before installing and starting SSH Tectia Server.

If you are familiar with SSH Tectia Server 4.x or older, we recommend that you read *SSH Tectia Client/Server Migration Guide*. It contains information on new and changed configuration options of SSH Tectia 5.0 and instructions for migrating existing installations of SSH Tectia 4.x to 5.0.

## 1.1 Component Terminology

The following terms are used throughout the documentation.

| | |
|---|---|
| client computer | The computer, typically a workstation, from which the Secure Shell connection is initiated. |
| host key | A public-key pair used as the identification of the Secure Shell server. |

| | |
|---|---|
| remote host | Refers to the other party of the connection, client computer or server computer, depending on the viewpoint. |
| Secure Shell client | A client-side application that uses the Secure Shell version 2 protocol, for example `sshg3`, `sftpg3` or `scpg3` of SSH Tectia Client, or SSH Tectia Connector. |
| Secure Shell server | A server-side application that uses the Secure Shell version 2 protocol. |
| server computer | The computer, typically a server, on which the Secure Shell service is running and to which the Secure Shell client is connected. |
| SFTP server | A server-side application that provides a secure file transfer service as a subsystem of the Secure Shell server. |
| SSH Tectia Client | A software component installed on a workstation. SSH Tectia Client provides secure interactive file transfer and terminal client functionality for remote users and system administrators to access and manage servers running SSH Tectia Server or other applications using the Secure Shell protocol. It also supports (non-transparent) static and dynamic tunneling of TCP-based applications. |
| SSH Tectia Client (F) | A software component installed on a workstation or a server. SSH Tectia Client (F) is especially designed for secure file transfer, in both interactive and automatic modes. |
| SSH Tectia client/server solution | The SSH Tectia client/server solution consists of three products, SSH Tectia Server, SSH Tectia Client, and SSH Tectia Connector. |
| SSH Tectia Connector | SSH Tectia Connector is a transparent end-user desktop client that provides dynamic tunneling of client/server connections without the need to re-configure the tunneled applications. It enables corporate end users to connect to business applications securely and automatically when an IP connection is established, while being fully transparent to the user. SSH Tectia Connector connects to SSH Tectia Server (T) and SSH Tectia Server (M). |
| SSH Tectia Server | SSH Tectia Server is a server-side component for SSH Tectia Connector and Client. There are four separate versions of the product available: SSH Tectia Server (A) for secure remote administration, SSH Tectia Server (F) for secure file transfer, SSH Tectia Server (T) for secure application connectivity, and SSH Tectia Server (M) for IBM mainframes. |
| SSH Tectia Server (A) | The administration (A) server is available for Linux, Unix, and Windows platforms. It is a Secure Shell server for SSH Tectia Client. |

| | |
|---|---|
| SSH Tectia Server (F) | The file transfer (F) server is available for Linux, Unix, and Windows platforms. It is a Secure Shell server for SSH Tectia Client and Client (F). |
| SSH Tectia Server (M) | The mainframe (M) server is available for IBM z/OS platforms. It is a Secure Shell server for SSH Tectia Connector, Client (F), and Client. |
| SSH Tectia Server (T) | The tunneling (T) server is available for Linux, Unix, and Windows platforms. It is a Secure Shell server for SSH Tectia Connector, Client (F), and Client. |
| tunneled application | TCP application secured by a Secure Shell connection. |

## 1.2 Documentation Conventions

The following special conventions are used in this document:

**Table 1.2. Documentation conventions**

| Convention | Usage | Example |
|---|---|---|
| **Bold** | Menus, GUI elements, strong emphasis | Click **Apply** or **OK**. |
| → | Series of menu selections | Select File → Save |
| `Monospace` | Filenames, commands, directories, URLs etc. | Refer to `readme.txt` |
| *Italics* | Reference to other documents or products, emphasis | See *SSH Tectia Client User Manual* |

### Note

Indicates neutral or positive information that emphasizes or supplements important points of the main text. Supplies information that may apply only in special cases (memory limitations, equipment configurations, specific versions of a program).

### Caution

Advises users that failure to take or avoid a specified action could result in loss of data.

## 1.3 Customer Support

If the product documentation does not answer all your questions, you can find the SSH Tectia FAQ and Knowledge Base at http://support.ssh.com/.

If you have purchased a maintenance agreement, you are entitled to technical support from SSH Communications Security. Review your agreement for specific terms.

Please see the following page for more information on submitting support requests, feature requests, or bug reports, and on accessing the available online resources: http://www.ssh.com/support/contact.

# Chapter 2 Installing SSH Tectia Server

This chapter contains instructions on installing (and removing) SSH Tectia Server on the supported Unix, Linux, Windows platforms.

## 2.1 Planning the Installation

This section lists the supported platforms and gives the necessary pre-requisites for the SSH Tectia Server installation.

### 2.1.1 System Requirements

The following operating systems are supported as SSH Tectia Server (A/F/T) platforms:

- IBM AIX 5L 5.1, 5.2, and 5.3 (POWER)
- HP-UX 11.00 and 11i v1 (11.11) (PA-RISC)
- HP-UX 11i v1.6 (11.22) and 11i v2 (11.23) (IA64)
- Red Hat Enterprise Linux 3 and 4 (x86, POWER*, zSeries*)
- SUSE LINUX Professional 9.1 and 9.2 (x86)
- SUSE LINUX Enterprise Server 9 (x86, POWER*, zSeries*)
- Sun Solaris 2.6, 7, 8, 9, and 10 (SPARC)
- Microsoft Windows 2000 with SP4, XP with SP1-SP2, and Server 2003 with SP1 (x86)

* SSH Tectia Server (T) only.

The following minimum hardware is required:

### SSH Tectia Server (A) and SSH Tectia Server (F)

- Any standard hardware
- 100 MB free disk space
- CD-ROM drive (*optional, if installed from network*)
- TCP/IP connection

### SSH Tectia Server (T)

- 1 GB RAM for hundreds of simultaneous tunnels
- 100 MB free disk space
- CD-ROM drive (*optional, if installed from network*)
- TCP/IP connection

## 2.1.2 Packaging

On Unix and Linux platforms, SSH Tectia Server comes in two installation packages. The first package contains the common components of SSH Tectia Client and Server. The second contains the specific components of SSH Tectia Server.

On Windows, SSH Tectia Server comes in a single MSI installation package.

SSH Tectia Server (A), Server (F), and Server (T) use the same installation package. The functionality of the server is controlled by the license file. See Section 2.1.3.

Table 2.1 summarizes the required SSH Tectia Server packages on different platforms.

**Table 2.1. The SSH Tectia Server installation packages**

| SSH Tectia Server (A/F/T) on Unix and Linux | SSH Tectia Server (A/F/T) on Windows |
|---|---|
| Common | Server |
| Server | |

## 2.1.3 Licensing

SSH Tectia Server requires a license file to function.

Each Server type (A, F, and T) uses a license file of its own. Depending on the platform and the Server type you have purchased, you have one of the following files:

- On Unix: `tectia_server_A_50.lic`, `tectia_server_F_50.lic`, or `tectia_server_T_50.lic`.

- On Windows: `sts50.dat` (the contents of the file is different with each SSH Tectia Server type).

In the CD-ROM, the license files can be found in the `install/<platform>` directory.

After installation, the license file should be located in `/etc/ssh2/licenses` on Unix and in `<INSTALLDIR>\SSH Tectia AUX\licenses` on Windows (the default installation directory is `C:\Program Files\SSH Communications Security\SSH Tectia`).

On Windows, when installing from the CD-ROM, the license file is automatically copied to the right directory. In other cases, the license file has to be copied manually.

### 2.1.4 Upgrading from Version 4.x to 5.0

On Unix and Linux platforms, earlier versions of SSH Tectia Server should be removed before installing SSH Tectia Server 5.0. (When installing via SSH Tectia Manager, this is handled automatically.)

On Windows, SSH Tectia Server 4.1 and later can be upgraded by installing a newer version of the software on top of the older version. SSH Tectia Server 4.0 and earlier use a different type of installation package and must be uninstalled before installing the new version.

The configuration file format and file locations have changed in SSH Tectia Server 5.0. The old configuration files form 4.x will not be used with 5.0, but they must be converted manually to the new format.

A separate document, *SSH Tectia Client/Server Migration Guide*, gives detailed instructions on upgrading from SSH Tectia client/server solution 4.x to SSH Tectia client/server solution 5.0, including information on migrating the configuration files.

**Note**

Back up all your configuration files before starting the upgrade.

### 2.1.5 Upgrading from Version 5.0

SSH Tectia Server can be upgraded from a previous 5.0.x installation to a later 5.0.x simply by installing the newer version of the software on top of the older version.

If installed on the same machine, SSH Tectia Client and SSH Tectia Server 5.0 should be always upgraded at the same time, because there are dependencies between the common components.

**Note**

Back up all your configuration files before starting the upgrade.

## 2.2 Installing the SSH Tectia Server Software

This section gives instructions on installing SSH Tectia Server locally on the supported operating systems. SSH Tectia Server can also be installed via SSH Tectia Manager. See *SSH Tectia Manager Administrator Manual* for more information.

### 2.2.1 Installing on AIX

On the CD-ROM, the installation packages for AIX 5L platforms are located in the `/install/aix/` directory. Two packages are required: one for the common components of SSH Tectia Client and Server, and another for the specific components of SSH Tectia Server.

**Note**

You need GNU `gzip` in order to install SSH Tectia Server on AIX.

To install SSH Tectia Server on AIX, do the following:

1. (*Not necessary in "third-digit" maintenance updates.*) Copy the license file to the `/etc/ssh2/licenses` directory. See Section 2.1.3.

   If this is the initial installation of SSH Tectia Server 5.x, the directory does not yet exist. You can either create it manually or copy the license after the installation. In the latter case, you have to start the server manually after copying the license file.

2. Unpack the packages using the following commands:

   ```
   $ gzip -d ssh-tectia-common-<ver>-aix5.x.bff.gz
   $ gzip -d ssh-tectia-server-<ver>-aix5.x.bff.gz
   ```

   In the commands, `<ver>` is the current package version of SSH Tectia Server (for example, `5.0.0.777`).

3. Install the packages by running the following commands with root privileges:

   ```
   # installp -d ssh-tectia-common-<ver>-aix5.x.bff SSHTectia.Common
   # installp -d ssh-tectia-server-<ver>-aix5.x.bff SSHTectia.Server
   ```

   The server host key is generated during the installation. Key generation may take several minutes on slower machines.

4. The installation should (re)start the server automatically.

   If the server does not start (because of a missing license, for example), you can start it after correcting the problem by issuing the command:

   ```
   # /opt/tectia/sbin/rc.ssh-server-g3 start
   ```

## 2.2.2 Installing on HP-UX

SSH Tectia Server is available for HP-UX 11.x on PA-RISC (`11.00`) and for HP-UX 11.x on Itanium (`11.22`).

SSH Tectia Server includes support for Entrust certificates on HP-UX 11.0. The necessary libraries are automatically included in the installation.

On the CD-ROM, the installation packages for HP-UX platforms are located in the `/install/hp-ux/` directory. Two packages are required: one for the common components of SSH Tectia Client and Server, and another for the specific components of SSH Tectia Server.

To install SSH Tectia Server on HP-UX, do the following:

1. (*Not necessary in "third-digit" maintenance updates.*) Copy the license file to the `/etc/ssh2/licenses` directory. See Section 2.1.3.

   If this is the initial installation of SSH Tectia Server 5.x, the directory does not yet exist. You can either create it manually or copy the license after the installation. In the latter case, you have to start the server manually after copying the license file.

2. Unpack the packages with `gunzip`. In order to be installable, the created packages must have the correct long file name:

   ```
   $ gunzip ssh-tectia-common-<ver>-sd-<hpuxver>.depot.gz
   $ gunzip ssh-tectia-server-<ver>-sd-<hpuxver>.depot.gz
   ```

   In the package name, `<ver>` is the current package version of SSH Tectia Server (for example, `5.0.0.777`) and `<hpuxver>` is the version of the HP-UX operating system (`11.00` or `11.22`).

3. Install the packages by running the following command with root privileges:

   ```
   # swinstall -s <path>/ssh-tectia-common-<ver>-sd-<hpuxver>.depot SSHG3common
   # swinstall -s <path>/ssh-tectia-server-<ver>-sd-<hpuxver>.depot SSHG3server
   ```

   In the command, `<path>` is the full path to the installation package (HP-UX requires this even when the command is run in the same directory).

   The server host key is generated during the installation. Key generation may take several minutes on slower machines.

4. The installation should (re)start the server automatically.

   If the server does not start (because of a missing license, for example), you can start it after correcting the problem by issuing the command:

   ```
   # /sbin/init.d/ssh-server-g3 start
   ```

### 2.2.3 Installing on Linux

SSH Tectia Server for Linux platforms is supplied in RPM (Red Hat Package Manager) binary packages. The RPMs are available for Red Hat and SUSE Linux running on Intel x86 (`i386`), on IBM POWER (`ppc64pseries`), and on IBM S/390 or zSeries (`s390x`) platforms. The package for the x86 architecture is compatible also with the 64-bit versions of Red Hat and SUSE Linux running on x86-64 platforms. The IBM Linux packages are only available with SSH Tectia Server (T).

On the installation CD-ROM, the installation packages for Linux are located in the `/install/linux/` directory. Two packages are required: one for the common components of SSH Tectia Client and Server, and another for the specific components of SSH Tectia Server.

To install SSH Tectia Server on Linux, do the following:

1. (*Not necessary in "third-digit" maintenance updates.*) Copy the license file to the `/etc/ssh2/licenses` directory. See Section 2.1.3.

   If this is the initial installation of SSH Tectia Server 5.x, the directory does not yet exist. You can either create it manually or copy the license after the installation. In the latter case, you have to start the server manually after copying the license file.

2. Install the packages with root privileges:

   ```
   # rpm -Uvh ssh-tectia-common-<ver>.<arch>.rpm
   # rpm -Uvh ssh-tectia-server-<ver>.<arch>.rpm
   ```

   In the commands, `<ver>` is the current package version of SSH Tectia Server (for example, `5.0.0.777`) and `<arch>` is the platform architecture (`i386`, `ppc64pseries`, or `s390x`).

   The server host key is generated during the installation. Key generation may take several minutes on slower machines.

3. The installation should (re)start the server automatically.

   If the server does not start (because of a missing license, for example), you can start it after correcting the problem by issuing the command:

   ```
   # /etc/init.d/ssh-server-g3 start
   ```

### 2.2.4 Installing on Solaris

SSH Tectia Server is available for Sun Solaris on the SPARC architecture.

SSH Tectia Server includes support for Entrust certificates on Solaris 7 and 8. The necessary libraries are automatically included in the installation.

On the CD-ROM, the installation packages for Solaris are located in the `/install/solaris/` directory. Two packages are required: one for the common components of SSH Tectia Client and Server, and another for the specific components of SSH Tectia Server.

To install SSH Tectia Server on Solaris, do the following:

1. (*Not necessary in "third-digit" maintenance updates.*) Copy the license file to the `/etc/ssh2/licenses` directory. See Section 2.1.3.

   If this is the initial installation of SSH Tectia Server 5.x, the directory does not yet exist. You can either create it manually or copy the license after the installation. In the latter case, you have to start the server manually after copying the license file.

2. Unpack the installation packages to a suitable place. The standard place is `/var/spool/pkg` in a Solaris environment.

   ```
   $ uncompress ssh-tectia-common-<ver>-sparc-solaris2.6-10.pkg.Z
   $ uncompress ssh-tectia-server-<ver>-sparc-solaris2.6-10.pkg.Z
   ```

   In the command, `<ver>` is the current package version of SSH Tectia Server (for example, `5.0.0.777`).

3. Then install the packages with the `pkgadd` tool with root privileges:

   ```
   # pkgadd -d ssh-tectia-common-<ver>-sparc-solaris2.6-10.pkg all
   # pkgadd -d ssh-tectia-server-<ver>-sparc-solaris2.6-10.pkg all
   ```

   The server host key is generated during the installation. Key generation may take several minutes on slower machines.

4. The installation should (re)start the server automatically.

   If the server does not start (because of a missing license, for example), you can start it after correcting the problem by issuing the command:

   ```
   # /etc/init.d/ssh-server-g3 start
   ```

## 2.2.5 Installing on Windows

The Windows installation packages are provided in the MSI (Microsoft Installer) format.

SSH Tectia Server includes support for Entrust certificates on Windows. The necessary libraries are automatically included in the installation.

The installation is carried out by a standard installation wizard. The wizard will prompt you for information and will copy the program files, install the services, and generate the host key pair for the server.

On the CD-ROM, the installation package for Windows is located in the `/install/windows/` directory.

**Note**

You must have administrator rights to install SSH Tectia Server on Windows.

To install SSH Tectia Server on Windows, do the following:

1. Locate the installation file `ssh-tectia-server-<ver>.msi` (where `<ver>` corresponds to the version and build number, for example `5.0.0.777`). Double-click the installation file, and the installation wizard will start.

2. Follow the wizard through the installation steps and fill in information as requested.

   The server host key is generated during the installation. Key generation may take several minutes on slower machines.

3. When the installation has finished, click **Finish** to exit the wizard.

4. The installation should (re)start the server automatically.

   If the server does not start (because of a missing license, for example), you can start it after correcting the problem from the Windows **Services** console. See Section 3.2.2

## 2.3 Removing the SSH Tectia Server Software

This section gives instructions on removing SSH Tectia Server from the supported operating systems.

### 2.3.1 Removing from AIX

To remove SSH Tectia Server from an AIX environment, do the following:

1. Stop SSH Tectia Server with the following command:

```
# /opt/tectia/sbin/rc.ssh-server-g3 stop
```

2. Remove the installation by issuing the following command with root privileges:

```
# installp -u SSHTectia.Server
```

3. If you want to remove also the components that are common with SSH Tectia Client, give the following command:

```
# installp -u SSHTectia.Common
```

**Note**

The uninstallation procedure removes only the files that were created when installing the software. Any configuration files and host keys have to be removed manually.

### 2.3.2 Removing from HP-UX

To remove SSH Tectia Server from an HP-UX environment, do the following:

1. Stop SSH Tectia Server with the following command:

   ```
   # /sbin/init.d/ssh-server-g3 stop
   ```

2. Remove the installation by issuing the following command with root privileges:

   ```
   # swremove SSHG3server
   ```

3. If you want to remove also the components that are common with SSH Tectia Client, give the following command:

   ```
   # swremove SSHG3common
   ```

**Note**

The uninstallation procedure removes only the files that were created when installing the software. Any configuration files and host keys have to be removed manually.

### 2.3.3 Removing from Linux

To remove SSH Tectia Server from a Linux environment, do the following:

1. Stop SSH Tectia Server with the following command:

   ```
   # /etc/init.d/ssh-server-g3 stop
   ```

2. Remove the installation by issuing the following command with root privileges:

   ```
   # rpm -e ssh-tectia-server-<ver>
   ```

   In the command, `<ver>` is the package version of SSH Tectia Server to be removed (for example, `5.0.0.777`).

3. If you want to remove also the components that are common with SSH Tectia Client, give the following command:

   ```
   # rpm -e ssh-tectia-common-<ver>
   ```

**Note**

The uninstallation procedure removes only the files that were created when installing the software. Any configuration files and host keys have to be removed manually.

### 2.3.4 Removing from Solaris

To remove SSH Tectia Server from a Solaris environment, do the following:

1. Stop SSH Tectia Server with the following command:

   ```
   # /etc/init.d/ssh-server-g3 stop
   ```

2. Remove the installation by issuing the following command with root privileges:

   ```
   # pkgrm SSHG3srvr
   ```

3. If you want to remove also the components that are common with SSH Tectia Client, give the following command:

   ```
   # pkgrm SSHG3cmmn
   ```

**Note**

The uninstallation procedure removes only the files that were created when installing the software. Any configuration files and host keys have to be removed manually.

### 2.3.5 Removing from Windows

To remove SSH Tectia Server from a Windows environment, do the following:

1. From the **Start** menu, open the Windows **Control Panel** and double-click **Add or Remove Programs**.
2. From the program list, select **SSH Tectia Server** and click **Remove**.
3. Click **Yes** to confirm.

**Note**

The uninstallation procedure removes only the files that were created when installing the software. Any configuration files and host keys have to be removed manually.

# Chapter 3 Getting Started

This chapter provides information on how to get started with SSH Tectia Server software after it has been successfully installed.

Before you proceed, please see the separate document *SSH Tectia Client/Server Product Description* for background information such as choosing the authentication method.

The default configuration of SSH Tectia Server is aimed toward usability. Chapter 6, Chapter 7, and Chapter 8 give advice on configuring the servers for their intended use.

## 3.1 Location of Installed Files

This section lists the locations of the installed executables, configuration files, key files, and the license file.

### 3.1.1 File Locations on Unix

On Unix platforms, the SSH Tectia Server files are located in the following directories:

- `/etc/ssh2`
  - `/etc/ssh2/ssh-server-config.xml` : server configuration file (see ssh-server-config(5))
  - `/etc/ssh2/hostkey` : default server host private key file
  - `/etc/ssh2/hostkey.pub` : default server host public key file (not required)
  - `/etc/ssh2/licenses` : license file directory (see Section 2.1.3)
  - `/etc/ssh2/ssh-tectia/auxdata/ssh-server-ng` : server configuration file DTD directory
- `/opt/tectia/sbin` : system binaries such as `ssh-server-g3`
- `/opt/tectia/bin` : user binaries such as `ssh-keygen-g3`

- `/opt/tectia/libexec` : library binaries
- `/opt/tectia/lib/sshsecsh` : library binaries

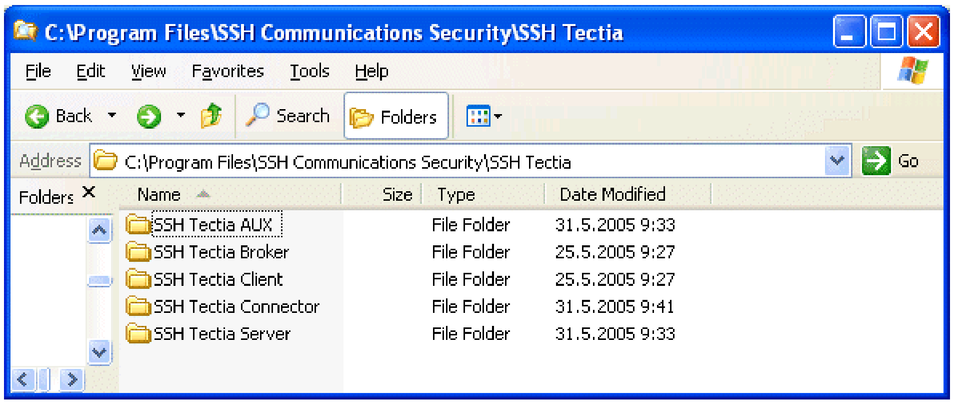
## 3.1.2 File Locations on Windows

On Windows, the default installation directory for SSH Tectia products is `C:\Program Files\SSH Communications Security\SSH Tectia\`.

On Windows, the SSH Tectia Server files are located in the following directories:

- `<INSTALLDIR>\SSH Tectia Server` : system binaries such as `ssh-server-g3.exe`
  - `<INSTALLDIR>\SSH Tectia Server\ssh-server-config.xml` : server configuration file (see ssh-server-config(5))
  - `<INSTALLDIR>\SSH Tectia Server\hostkey` : default server host private key file
  - `<INSTALLDIR>\SSH Tectia Server\hostkey.pub` : default server host public key file (not required)
- `<INSTALLDIR>\SSH Tectia AUX` : auxiliary binaries such as `ssh-keygen.exe`
  - `<INSTALLDIR>\SSH Tectia AUX\ssh-server-ng` : server configuration file DTD directory
  - `<INSTALLDIR>\SSH Tectia AUX\licenses` : license file directory (see Section 2.1.3)

Figure 3.1 shows the SSH Tectia directory structure when also SSH Tectia Client and SSH Tectia Connector have been installed on the same machine.


C:\Program Files\SSH Communications Security\SSH Tectia
File
Edit
View
Favorites
Tools
Help
Back
Search
Folders
Address
C:\Program Files\SSH Communications Security\SSH Tectia
Go
Folders

| Name | Size | Type | Date Modified |
|---|---|---|---|
| SSH Tectia AUX | | File Folder | 31.5.2005 9:33 |
| SSH Tectia Broker | | File Folder | 25.5.2005 9:27 |
| SSH Tectia Client | | File Folder | 25.5.2005 9:27 |
| SSH Tectia Connector | | File Folder | 31.5.2005 9:41 |
| SSH Tectia Server | | File Folder | 31.5.2005 9:33 |

**Figure 3.1. The SSH Tectia directory structure on Windows**

## 3.2 Starting and Stopping the Server

SSH Tectia Server is started automatically after installation and at boot time.

## 3.2.1 Starting and Stopping on Unix

If the server needs to be started, stopped, or restarted manually, the `ssh-server-g3` script can be used.

The command has the following syntax:

```
ssh-server-g3 [command]
```

The `command` can be either `start`, `stop`, `restart`, or `reload`.

**start**
: Start the server.

**stop**
: Stop the server. Existing connections stay open until closed from the client side.

**restart**
: Start a new server process. Existing connections stay open using the old server process. The old process is closed after the last old connection is closed from the client side.

**reload**
: Reload the configuration file. Existing connections stay open.

The path to the `ssh-server-g3` script is different on each operating system:

- On AIX:

  ```
  # /opt/tectia/sbin/rc.ssh-server-g3 [command]
  ```

- On Linux and Solaris:

  ```
  # /etc/init.d/ssh-server-g3 [command]
  ```

- On HP-UX:

  ```
  # /sbin/init.d/ssh-server-g3 [command]
  ```

## 3.2.2 Starting and Stopping on Windows

To start and stop SSH Tectia Server on Windows, use the **Services** console:

1. From the **Start** menu, open the Windows **Control Panel** and double-click **Administrative Tools**.
2. Double-click **Services**. The **Services** console opens.
3. From the list, right-click on SSH Tectia Server. From the shortcut menu, you can now **Start**, **Stop**, **Pause**, **Resume**, or **Restart** the server.

If the server is paused, the existing connections will stay open but the server will not accept new connections.

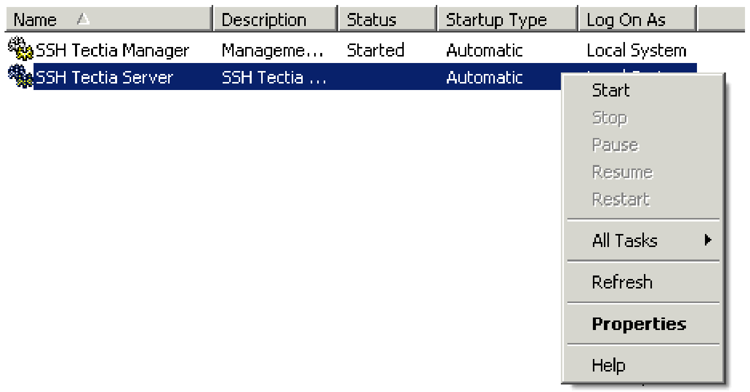


**Figure 3.2. Starting and stopping SSH Tectia Server from the Windows Services console**

## 3.3 Examples of Use

For examples of using SSH Tectia Server, see the Compatibility Note documents at http://www.ssh.com/products/material/compatability/.

# Chapter 4 Configuring SSH Tectia Server

SSH Tectia Server uses an XML-based configuration file `ssh-server-config.xml` that allows flexible implementation of real-life enterprise security policies.

The configuration file follows the same logic that is used in setting up a Secure Shell connection:

1. The client initiates a connection to the server.
2. The server checks whether connections from the client address are allowed. The client and server perform key exchange where the server authenticates itself to the client, and ciphers and MACs are selected for the connection.
3. The server requests the user to authenticate itself to the server. The server may offer a selection of authentication methods, or require several authentication methods to be passed in succession.
4. The server determines the services the client is allowed to use.

The configuration file can be edited with an ASCII text editor or an XML editor (see ssh-server-config(5)). On Windows, you can use the included **SSH Tectia Server Configuration** tool (see Section 4.1).

## ssh-server-config

ssh-server-config -- SSH Tectia Server configuration file format

The SSH Tectia Server configuration file `ssh-server-config.xml` is a valid XML file.

On Unix, the file is stored in the `/etc/ssh2/` directory. The XML DTD can be found in the `/etc/ssh2/ssh-tectia/auxdata/ssh-server-ng` directory.

On Windows, the file is stored by default in the `C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Server` directory. The XML DTD can be found in the `C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia AUX\ssh-server-ng` directory.

If the configuration file cannot be found or some of the elements are missing, hardcoded default values are used. The default values are specified in the `ssh-server-config-default.xml` file in the same directory with the configuration file. However, this default configuration file is not actually read. It only shows the hardcoded system defaults.

In addition, an example configuration file `ssh-server-config-example.xml` can be found in the same directory with the configuration file. This file contains useful examples of the various configuring options.

The configuration file is divided in four blocks:

- general server parameters (`params`)
- connection rules (`connections`)
- authentication methods (`authentication-methods`)
- allowed services (`services`)

In the `connections` and `authentication-methods` blocks, different *selectors* can be used to set access rules to users based on the user parameters such as username or location. Users can be divided to groups dynamically, for example, based on the authentication method they used to log in. In the `services` block, each group can then be allowed or denied services such as tunneling, file transfer, and terminal access.

The configuration file is read in top-down order during connection setup. If a connection is denied in one of the blocks, the connection setup phase ends immediately and the rest of the configuration file is not read.

This section describes the options available in the SSH Tectia Server configuration file. See Appendix B for more information on the syntax of the configuration file.

## Selectors

The connection settings can be changed based on *selectors* in the configuration file. Using selectors it is possible, for example, to:

- allow/deny connections from certain IP addresses
- require different authentication methods based on username or group
- restrict access based on a certificate field

Selectors are used in the *Connections*, *Authentication Methods*, and *Services* elements.

Selectors match to *blackboard fields*.

The most commonly used selector parameters have their own element (`interface`, `certificate`, `host-certificate`, `ip`, `user`, `user-group`, and `user-privileged`).

Uncommon parameters can be used with the generic `blackboard` element by giving the parameter as a value of the `field` attribute.

When a parent element contains multiple child elements with selectors, the first functional child element matched is used, and the rest are ignored. For example, if the `connections` element has multiple `connection` child elements, but the first one has an empty selector, or no selectors at all, that `connection` element always matches and the remaining are never used.

## Wild Cards in Selectors

Simple wild cards can be used inside selector values.

An asterisk (`*`) matches any string. A question mark (`?`) matches any character. A dash (`-`) can be used to specify a range.

For example, the following selector matches to usernames `jdoe` and `jdox`, but not to `jdoel`:

```
<selector>
  <user name="jdo?" />
</selector>
```

For example, the following selector matches to IP addresses between `192.168.0.42` and `192.168.0.82`:

```
<selector>
  <ip address="192.168.0.42-192.168.0.82" />
</selector>
```

## Selector Handling Rules

Selector elements are in an OR relation (one of the selectors must match for the parent element to match). For example, the following block matches if either the IP address is `192.168.0.3` or the user ID is `11001001`:

```
<selector>
  <ip address="192.168.0.3" />
</selector>
<selector>
  <user id="11001001" />
</selector>
```

Selector items in the same selector element are normally in an AND relation (both selectors must match for the element to match). For example, the following element matches if both the IP address is `192.168.0.3` and the user ID is `11001001`:

```
<selector>
  <ip address="192.168.0.3" />
  <user id="11001001" />
</selector>
```

However, selector items in the same selector element matching to the same blackboard field are in an OR relation to each other. The following three examples produce the same result, either the username `exa` or `mple` matches:

```
<selector>
  <user name="exa" />
  <user name="mple" />
</selector>

<selector>
  <user name="exa" />
</selector>
<selector>
  <user name="mple" />
</selector>

<selector>
  <user name="exa,mple" />
</selector>
```

An empty selector always matches:

```
<selector />
```

Also, typically, if an element accepts selectors, but none are given, the element is assumed to have an empty selector, which will then always match.

## Document Type Declaration and the Root Element

The server configuration file is a valid XML file and starts with the Document Type Declaration.

The root element in the configuration file is `secsh-server`. It can include `params`, `connections`, `authentication-methods`, and `services` elements.

An example of an empty configuration file is shown below:

```
<!DOCTYPE secsh-server SYSTEM
    "../auxdata/ssh-server-ng/ssh-server-ng-config-1.dtd">

<secsh-server>

  <params />
  <connections>
    <connection />
  </connections>
  <authentication-methods />
  <services>
    <rule />
  </services>
```

```
</secsh-server>
```

### Note

It is not mandatory to include all elements in the configuration file. If an element is missing the default values shown in the `ssh-server-config-default.xml` file are used. However, the configuration file should always contain the `DOCTYPE` declaration. If the Document Type Declaration is omitted, the server will not be able to parse the configuration properly.

## The `params` Element

The `params` element defines the general server parameters, such as the location of the host key file, the listen address, logging, connection limits, and certificate validation settings.

**crypto-lib**

This element selects the cryptographic library mode to be used. Either the standard version (`standard`) or the FIPS 140-2 certified version (`fips`) of the crypto library can be used. The library name is given as a value of the `mode` attribute. By default, standard crypto libraries are used.

```
<crypto-lib mode="standard" />
```

**settings**

This element contains miscellaneous settings. It has four attributes: `proxy-scheme`, `xauth-path`, `ignore-aix-rlogin`, and `user-config-dir`.

The `proxy-scheme` attribute defines rules for HTTP or SOCKS proxy servers that SSH Tectia Server uses when a client forwards a connection (local tunnel).

The format of the attribute value is a sequence of rules delimited by semicolons (`;`). Each rule has a format that resembles the URL format. In a rule, the connection type is given first. The type can be `direct`, `socks`, `socks4`, `socks5`, or `http-connect` (`socks` is a synonym for `socks4`). This is followed by the server address and port. If the port is not given, the default ports 1080 for SOCKS and 80 for HTTP are used.

After the address, zero or more conditions delimited by commas (`,`) are given.

```
direct:///[cond[,cond]...]
socks://server/[cond[,cond]...]
socks4://server/[cond[,cond]...]
socks5://server/[cond[,cond]...]
http-connect://server/[cond[,cond]...]
```

The IP address/port conditions have an address pattern and an optional port range:

```
ip_pattern[:port_range]
```

The `ip_pattern` may have one of the following forms:

- IP: a single IP address
- FROM-TO: an IP address range from FROM to TO
- IP/MASK: a sub-network with a network mask with MASK significant bits.

DNS name conditions consist of a hostname which may be a regular expression containing the characters "*" and "?" and a port range:

```
name_pattern[:port_range]
```

An example `proxy-scheme` is shown below. It causes the server to access the callback address and the `ssh.com` domain directly, access `*.example` with HTTP CONNECT, and all other destinations with SOCKS4.

```
"direct:///127.0.0.0/8,*.ssh.com;
 http-connect://http-proxy.ssh.com:8080/*.example;
 socks://fw.ssh.com:1080/"
```

The `xauth-path` attribute contains a path to a supplementary XAuth binary used with X11 forwarding on Unix platforms.

The `ignore-aix-rlogin` attribute defines whether the server should ignore the remote login restriction on AIX. Local login permission is still honored. The value must be `yes` or `no`. The default is `no`.

The `user-config-dir` attribute can be used to specify where user-specific configuration data is found. With this, the administrator can control those options that are usually controlled by the user. This is given as a pattern string which is expanded by SSH Tectia Server. In the string, `%D` is the user's home directory, `%U` is the user's login name, `%IU` is the user's user ID (uid), and `%IG` is user's group ID (gid).

```
<settings
 proxy-scheme="direct:///10.0.0.0/8,localhost;socks5://fw.example.com:1080/"
 xauth-path="$XAUTH_PATH"
 ignore-aix-rlogin="no"
 user-config-dir="%D/.ssh2" />
```

**hostkey**

The `hostkey` element defines the location of the private host key and optionally the location of the public key and/or certificate. The elements inside the element must be given in the right order (private key before public).

Inside one `hostkey` element either the public key or the certificate can be given, not both.

Giving the public key in the configuration file is not mandatory. It will be derived from the private key if it is not found otherwise. However, specifying the public key will decrease the start-up time for the software, as deriving the public key is a fairly slow operation.

**private**

This element gives the path to the private key file as a value of the `file` attribute.

The key file should be located on a local drive. Network or mapped drives should not be used, as the server program may not have proper access rights for them. The default is `hostkey`, in the directory `/etc/ssh2` on Unix and `C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Server` on Windows. The key file should be readable only by the server itself.

**public**

This element gives the path to the public key file as a value of the `file` attribute.

The key file should be located on a local drive. Network or mapped drives should not be used, as the server program may not have proper access rights for them.

Alternatively, the public key can be specified as a base64-encoded ASCII element.

**x509-certificate**

This element gives the path to the X.509 user certificate file as a value of the `file` attribute.

Alternatively, the certificate can be specified as a base64-encoded ASCII element.

**externalkey**

This element defines an external host key. The `type` must be given as an attribute. The `init-info` can also be given.

Sample `hostkey` elements are shown below:

```
<hostkey>
  <private file="/etc/ssh2/hostkey_dsa" />
  <public file="/etc/ssh2/hostkey_dsa.pub" />
</hostkey>

<hostkey>
  <private file="/etc/ssh2/hostcert_rsa" />
  <x509-certificate file="/etc/ssh2/hostcert_rsa.crt" />
</hostkey>
```

**listener**

This element is used to specify where the server listens for connections. The element has three attributes: `id`, `address`, and `port`.

The `id` must be given as an attribute and it must be unique. Also the `port` and `address` can be given. The default port for listeners is 22.

Several listeners can be created to the same IP address to different ports. Each must have an unique ID.

Sample `listener` elements are shown below:

```
<listener id="internet" address="172.56.34.1" />
<listener id="intranet" address="10.0.0.1" />
<listener id="admin-private" port="222" />
```

**logging**

This element changes the logging settings that define the log event severities and logging facilities. The element contains one or more `log-events` elements.

**log-events**

This element sets the severity and facility of different logging events. The events have reasonable default values, which are used if no explicit logging settings are made. This setting allows customizing the default values.

For the events, `facility` and `severity` can be set as attributes. The events itself should be listed inside the `log-events` element.

The facility can be `normal`, `daemon`, `user`, `auth`, `local0`, `local1`, `local2`, `local3`, `local4`, `local5`, `local6`, `local7`, or `discard`. Setting the facility to `discard` causes the server to ignore the specified log events.

On Windows, only the `normal` and `discard` facilities are used.

The severity can be `informational`, `notice`, `warning`, `error`, `critical`, `security-success`, or `security-failure`.

Any events that are not specifically defined in the configuration file use the default values. The defaults can be overridden for all remaining events by giving an empty `log-events` element after all other definitions and setting a severity value for it.

For a complete list of log events, see Appendix D.

The following example sets the facility of the `connect`, `login`, and `logout` events to `daemon` and the severity to `notice`. It also sets the facility of the `filexfer_start` and `filexfer_end` events to `discard` (the events will not be logged). All other events use the default settings.

```
<logging>
  <log-events facility="daemon" severity="notice">
     connect login logout
  </log-events>
  <log-events facility="discard">
    filexfer_start filexfer_end
  </log-events>
</logging>
```

**limits**

This element sets the maximum number of connections and processes the server will handle. SSH Tectia Server uses a distributed architecture where the master server process launches several servant server processes that handle the actual connections.

The `max-processes` attribute defines the maximum number of servant processes the master server will launch. The `max-connections` attribute defines the maximum number of client connections allowed by each servant.

A value of `0` (zero) means that the number of connections or processes uses default values that depend on the operating system. By default, SSH Tectia Server will start with some number of servant processes running.

This setting is useful in systems with low resources. The server has to be restarted to use the changed setting.

A sample `limits` element is shown below:

```
<limits max-connections="256" max-processes="40" />
```

**cert-validation**

This element contains the CA certificates used in validation of the host-based and public-key authentication certificates. The element can have two attributes: `http-proxy-url` and `socks-server-url`.

The `http-proxy-url` attribute defines a HTTP proxy and the `socks-server-url` attribute defines a SOCKS proxy for making LDAP or OCSP queries for certificate validity.

The address of the proxy is given as the value of the attribute. The format of the address is `socks://username@socks_server:port/network/netmask,network/netmask ...` (with a SOCKS proxy) or `http://username@proxy_server:port/network/netmask,network/netmask ...` (with an HTTP proxy).

For example, by setting `socks-server-url` to `"socks://mylogin@socks.ssh.com:1080/203.123.0.0/16,198.74.23.0/24"`, the host `socks.ssh.com` and port `1080` are used as your SOCKS server for connections outside of networks `203.123.0.0` (16-bit domain) and `198.74.23.0` (8-bit domain). Those networks are connected directly.

The `cert-validation` element can contain multiple `ldap-server` and `ocsp-responder` elements, a `cert-cache-file` and `crl-auto-update` element, multiple `crl-prefetch` elements, a `dod-pki` element, and multiple `ca-certificate` elements.

The validity of a received certificate is checked separately using each of the defined `ca-certificate` elements in turn until they are exhausted (in which case the authentication fails), or a positive result is achieved. If the certificate is valid, the `connections` and `authentication-methods` elements determine whether the certificate allows the user to log in (of course, the correct signature generated by a matching private key is always required in addition to everything else).

**ldap-server**

This element specifies an LDAP server `address` and `port` used for fetching CRLs. Several LDAP servers can be specified by using several `ldap-server` elements.

CRLs are automatically retrieved from the CRL distribution point defined in the certificate to be checked if the point exists. Otherwise, the LDAP servers specified by these elements are used.

The default value for `port` is `389`.

**ocsp-responder**

This element specifies an OCSP (Online Certificate Status Protocol) responder service address in URL format (`url`). Several OCSP responders can be specified by using several `ocsp-responder` elements.

If the certificate has a valid `Authority Info Access` extension with an OCSP Responder URL, it is used instead of this setting. Note that for the OCSP validation to succeed, both the end-entity certificate and the OCSP Responder certificate must be issued by the same CA.

The `validity-period` in seconds for the OCSP data can be optionally defined. During this time, new OCSP queries for the same certificate are not made but the old result is used.

**cert-cache-file**

This element specifies the name of the `file` where the certificates and CRLs are stored when the SSH Tectia Server service is stopped, and read back in when the service is restarted.

**crl-auto-update**

This element turns on the CRL auto-update feature. When it is on, SSH Tectia Server periodically tries to download the new CRL before the old one has expired. The `update-before` attribute can be used to specify how many seconds before the expiration the update takes place. The `minimum-interval` sets a limit for the maximum update frequency (the default minimum interval is 30 seconds).

**crl-prefetch**

This element instructs SSH Tectia Server to periodically download a CRL from the specified `url`. The `interval` attribute specifies how often the CRL is downloaded. The default is `3600` (seconds).

**dod-pki**

This element defines whether the certificates are required to be compliant with the DoD PKI (US Department of Defense Public-Key Infrastructure). In practice, this means that the Digital Signature bit must be set in the Key Usage of the certificate. The `enable` attribute can have a value of `yes` or `no`. The default is `no`.

**ca-certificate**

This element enables user authentication using certificates. It can have four attributes: `name`, `file`, `disable-crls`, and `use-expired-crls`.

The `name` attribute must contain the name of the CA.

The element must either contain the path to the X.509 CA certificate file as a value of the `file` attribute, or include the certificate as a base64-encoded ASCII element.

CRL checking can be disabled by setting the `disable-crls` attribute to `yes`. The default is `no`.

**Note**

CRL usage should only be disabled for testing purposes. Otherwise it is highly recommended to always use CRLs.

Expired CRLs can be used by setting a numeric value (in seconds) for the `use-expired-crls` attribute. The default is 0 (do not use expired CRLs).

A sample `cert-validation` element is shown below:

```
<cert-validation http-proxy-url="http://proxy.example.com:800">
  <ldap-server address="ldap.example.com" port="389" />
  <ocsp-responder validity-period="60" url="https://ca.example.com/ocsp-1" />
  <cert-cache-file file="/var/cert-cache.dat" />
  <crl-auto-update update-before="30" minimum-interval="600" />
  <crl-prefetch interval="1800" url="http://ca.example.com/default.crl" />
  <dod-pki enable="no" />
  <ca-certificate name="exa-ca1" file="/etc/ssh2/exa-ca1.crt" />
  <ca-certificate name="exa-ca2" file="/etc/ssh2/exa-ca2.crt"
     use-expired-crls="3600" />
  <ca-certificate name="testonly-ca" file="/etc/ssh2/testonly-ca.crt"
     disable-crls="yes" />
</cert-validation>
```

## The `connections` Element

The `connections` element defines the basic rules for allowing and denying connections. The `connections` element includes one or more `connection` elements.

**connection**

Each `connection` element specifies either an allow or deny rule for connections. The element can have three attributes: `name`, `action`, and `tcp-keepalive`.

The word `allow` or `deny` is given as a value of the `action` attribute. By default (if the `action` attribute is omitted), the connection is allowed.

The `name` attribute can be used to give an identifier to the connection rule. This can be used, for example, in auditing.

The `tcp-keepalive` attribute defines whether the system should send keepalive messages to the other side. If they are sent, a broken connection or crash of one of the machines will be properly noticed. However, this means that connections will die if the route is down temporarily, and this can be annoying in some situations. On the other hand, if keepalive messages are not sent, sessions may hang indefinitely on the server, leaving "ghost" users and consuming server resources. The value must be `yes` or `no`. The default is `no` (do not send keepalives).

The `connection` element can include one or more selectors, a rekey setting, and cipher and MAC definitions.

**selector**

The selectors define to which connections this connection rule applies to. Only the `interface` and `ip` selectors can be used in the `connections` element. Other blackboard fields, for example the username, are not yet available at this stage of the connection. See the section called "Selectors".

**interface**

This selector matches to the listener interface `id` or `address` and/or `port`. At least one attribute must be given. If the `id` is defined, the others MUST NOT be given. If the `id` is not defined, either or both of `address` and `port` may be given.

**ip**

This selector matches to an IP `address` or `fqdn` (fully qualified domain name) of the client. Either `address` or `fqdn` can be given, not both.

The `address` can be in one of the following formats:

- an IP address range of the form `x.x.x.x-y.y.y.y`
- an IP mask of the form `x.x.x.x/y`
- a single IP address `x.x.x.x`

The `fqdn` attribute matches to an FQDN pattern (case-insensitive). The attribute can include a comma-separated list of allowed FQDN patterns.

**rekey**

This element specifies the number of `seconds` or transferred `bytes` after which the key exchange is done again.

If a value for both `seconds` and `bytes` is specified, rekeying is done whenever one of the values is reached, after which the counters are reset.

The defaults are `3600` seconds (1 hour) and `1000000000` bytes (~1 GB). The value `0` (zero) turns rekey requests off. This does not prevent the client from requesting rekeys.

**cipher**

This element selects a cipher `name` allowed by the server for data encryption. The supported ciphers are `3des-cbc`, `aes128-cbc`, `aes192-cbc`, `aes256-cbc`, `arcfour`, `blowfish-cbc`, `twofish-cbc`, `twofish128-cbc`, `twofish192-cbc`, `twofish256-cbc`, `crypticore128@ssh.com`, `seed-cbc@ssh.com`, and `none` (no encryption).

In the FIPS mode, the supported ciphers are `3des-cbc`, `aes128-cbc`, `aes192-cbc`, and `aes256-cbc`.

Multiple ciphers can be specified by using multiple `cipher` elements. The server allows `3des`, `aes128-cbc`, and `crypticore128@ssh.com` by default.

**mac**

This element selects a MAC `name` allowed by the server for data integrity verification. The supported MAC algorithms are `hmac-md5`, `hmac-md5-96`, `hmac-sha1`, `hmac-sha1-96`, `crypticore-mac@ssh.com`, and `none` (no data integrity verification).

In the FIPS mode, only `hmac-sha1` is supported.

Multiple MACs can be specified by using multiple `mac` elements. The server allows `hmac-md5` and `hmac-sha1` by default.

A sample `connection` element that allows connections from a specified IP address range is shown below:

```
<connection name="conn1" action="allow" tcp-keepalive="yes">
  <selector>
    <ip address="192.168.0.42-192.168.0.82" />
  </selector>
  <rekey seconds="600" bytes="500000000" />
  <cipher name="crypticore128@ssh.com" />
  <mac name="crypticore-mac@ssh.com" />
</connection>
```

A sample `connection` element that denies all connections is shown below. As the element does not contain any selectors, it matches always. This can be used as the last element in the `connections` element to deny all connections that were not explicitly allowed by the previous elements. (By default, non-matching connections would be allowed.)

```
<connection action="deny" />
```

## The `authentication-methods` Element

The `authentication-methods` element defines the authentication methods that are allowed and required by the server. It can have one attribute: `login-grace-time`. It can contain a `banner-message` and a `auth-file-modes` element and multiple `authentication` elements.

The `login-grace-time` attribute is used to specify a time after which the server disconnects if the user has not successfully logged in. If the value is set to `0`, there is no time limit. The default is `600` (seconds).

The `authentication` elements that are on the same level are considered allowed (one of them must succeed).

Several authentication methods can be set as required by nesting the `authentication` elements inside each other.

If the `authentication-methods` element does not exist, the server allows by default GSSAPI, public-key, keyboard-interactive, and password authentication (one of them must succeed).

However, if you put an empty `authentication-methods` element in the `ssh-server-config.xml` file, all users will be allowed to log in without authentication. Likewise, if you leave empty `authentication` elements inside `authentication-methods`, the matching users (everyone, if no selectors are used) will be allowed to log in without authentication.

### Caution

Do not leave the `authentication-methods` element empty in the `ssh-server-config.xml` file or else all users are allowed to log in without authentication. Either remove the element completely (to use the default authentication methods), or add `authentication` elements with authentication methods of your choice.

**banner-message**

This element specifies the path to the message that is sent to the client before authentication. The path is given as a value of the `file` attribute. Alternatively, the banner message can be given as the contents of the `banner-message` element.

Note, however, that the client is not obliged to show this message.

```
<banner-message file="/etc/ssh2/banner-message">
  This is the server banner message. If file attribute is set, this
  inlined text is ignored, and the file is read instead
  (like in this example).
</banner-message>
```

**auth-file-modes**

This element specifies whether SSH Tectia Server on Unix platforms should check permissions and ownership of the user's key files used for public-key authentication.

The word `yes` or `no` is given as a value of the `strict` attribute. If set to `yes`, the permissions and ownership of the `.ssh2` directory, the `.ssh2/authorization` file (if used), the `.ssh2/authorized_keys` directory (if used), and the keys listed in the `authorization` file or present in the `authorized_keys` directory are checked.

This is normally desirable because novices sometimes accidentally leave their directory or files world-writable, in which case anyone can edit the authorization and key files. The default is `yes`.

The `mask-bits` attribute can be used to specify the forbidden permission bits in octal format. The default is `022` (group and others must not have the write permission).

The ownership of the checked files and directories must be either root or the user.

```
<auth-file-modes strict="yes" mask-bits="022" />
```

**authentication**

Each `authentication` element specifies a chain of authentication methods. It can include one or more selectors and different authentication methods. It may also include other `authentication` elements.

Nesting `authentication` elements within each other sets the child methods as *required* (all must be passed for the authentication to be successful). Setting multiple `authentication` elements at the same level sets them as *optional* (one of the methods must be passed for the authentication to be successful).

The methods are read in order. For methods on the same level, the first matching method is used and the remaining methods are ignored. If the method has nested child methods, they are matched next using the same procedure.

The word `allow` or `deny` can be given as a value of the `action` attribute of the `authentication` element. By default, authentication is allowed.

If an authentication chain ends in a `deny` action, the user is not allowed to log in. In a nested chain of `authentication` elements, it is possible, for example, to set the parent method to deny authentication and a child element with a selector to allow authentication. If the user matches the selector and successfully completes the authentication methods, login is allowed.

The `authentication` element can additionally take a `set-group` attribute, which sets a group for the users that pass the particular authentication chain. The group definition can be later used in the `services` element.

If `set-group` is used here, it overrides any `group` definitions in the `services` element. See the section called "The `services` Element".

The authentication `name` can be optionally given as an attribute. This can be used, for example, in auditing.

## Caution

Do not leave an `authentication` element empty in the `ssh-server-config.xml` file. It will allow the matching users (everyone, if no selectors are used) to log in without authentication.

**selector**

The selectors define to which connections this authentication method applies to. All selectors can be used in the `authentication-methods` element. See the section called "Selectors" and the section called "The `connections` Element".

**certificate**

This selector matches to a `pattern` in a specified `field` of the user certificate. Using this selector requires that the authentication chain contains an `auth-publickey` element.

The field can be either `ca-list`, `issuer-name`, `subject-name`, `serial-number`, `altname-email`, `altname-upn`, `altname-ip`, or `altname-fqdn`.

The format of the pattern depends on the type of the field. The `ca-list` field contains a list of CA names separated by commas. The names that are defined in the `ca-certificate` element are used. The `issuer-name` and `subject-name` fields contain distinguished names, `serial-number` a positive integer. The `altname-fqdn` field contains a hostname and `altname-ip` an IP address or a range. The `altname-email` field contains an email address and `altname-upn` the principal name.

The `altname-fqdn`, `altname-upn`, `altname-email`, `subject-name`, and `issuer-name` selectors may contain the `%username%` keyword which is replaced with the user's name before comparing with the actual certificate data. The `%hostname%` keyword can be used in the same way and it is replaced by the client's FQDN. These patterns may also contain "*" and "?" globbing characters.

Patterns are normally matched case-insensitively. Alternatively, the pattern can be specified using the `pattern-case-sensitive` attribute.

**host-certificate**

This selector matches to a `pattern` in a specified `field` of the client host certificate.

The field can be either `ca-list`, `issuer-name`, `subject-name`, `serial-number`, `altname-email`, `altname-upn`, `altname-ip`, or `altname-fqdn`.

Patterns are normally matched case-insensitively. Alternatively, the pattern can be specified using the `pattern-case-sensitive` attribute.

**interface**

This selector matches to the listener interface `id` or `address` and/or `port`.

**ip**

This selector matches to an IP `address` or `fqdn` (fully qualified domain name) of the client.

**user**

This selector matches to a user `name` or `id`. A list of usernames or IDs can be given as a comma-separated list.

Names are normally matched case-insensitively. Alternatively, the names can be specified using the `name-case-sensitive` attribute.

**user-group**

This selector matches to a user group `name` or `id`. A list of user-group names or IDs can be given as a comma-separated list.

Names are normally matched case-insensitively. Alternatively, the names can be specified using the `name-case-sensitive` attribute.

**user-privileged**

This selector matches to a privileged user (administrator or root) or to a non-privileged user. The `value` can be `yes` (match to a privileged user) or `no` (match to a normal user).

**blackboard**

This selector matches to a `pattern` in a specified blackboard `field`.

Patterns are normally matched case-insensitively. Alternatively, the pattern can be specified using the `pattern-case-sensitive` attribute.

**publickey-passed**

This selector matches if authentication is passed using a normal public key (without a certificate). Using this selector requires that the authentication chain contains an `auth-publickey` element.

Optionally, the `length` range of the public key can be given as an attribute, for example `"1024-2048"` (keys from 1024 to 2048 bits match). The range can also be left open, for example `"1536-"` (keys over 1536 bits match).

**user-password-change-needed**

This selector matches on Unix platforms if the user password has expired and should be changed. The actual password change can be done by defining a rule with a forced `passwd` command. See the section called “The `services` Element” for an example.

**auth-publickey**

This element sets the public-key authentication method.

**auth-hostbased**

This element sets the host-based authentication method. The element can have one attribute: `require-dns-match`.

If the `require-dns-match` attribute is set to `yes`, host-based authentication will require the hostname given by the client to match the one found in DNS. If the hostname does not match, the authentication will fail. The default is `no` (exact match not required).

**auth-password**

This element sets the password authentication method.

The delay between failed attempts in seconds (`failure-delay`) and the maximum number of attempts (`max-tries`) can be given as attributes. The default delay is 2 seconds and default maximum is 3 attempts.

**auth-keyboard-interactive**

This element sets the keyboard-interactive authentication method.

The delay between failed attempts in seconds (`failure-delay`) and the maximum number attempts (`max-tries`) can be given as attributes. The default delay is 2 seconds and default maximum is 3 attempts.

The keyboard-interactive submethods are given as child elements. The supported methods are `submethod-password`, `submethod-pam`, `submethod-securid`, and `submethod-radius`, and `submethod-generic`.

**submethod-pam**

This element sets the keyboard-interactive PAM submethod in use. PAM is supported on Unix platforms.

The `dll-path` to the PAM DLL can be given in an attribute. The path must point to the operating-system specific PAM module, for example, `"/lib/libpam.so.1"` on Red Hat Linux.

**submethod-password**

This element sets the keyboard-interactive password submethod in use.

**submethod-securid**

This element sets the keyboard-interactive SecurID submethod in use.

The `dll-path` to the SecurID DLL can be given in an attribute. The path must point to the operating-system specific SecurID module, for example, `"/usr/lib/libaceclnt.so"` on Solaris.

**submethod-radius**

This element sets the keyboard-interactive RADIUS submethod in use.

The element can contain multiple `radius-server` child elements.

**radius-server**

This element defines a RADIUS server. The element has four attributes: `address`, `port`, `timeout`, and `client-nas-identifier`.

The `address` is the IP address of the RADIUS server. The `port` is the RADIUS server port. The default port is `1812`.

The `timeout` is the time in seconds after which the RADIUS query is terminated if no response is gained. The default is `10` seconds.

The `client-nas-identifier` defines the default NAS identifier to be used when talking to the RADIUS server.

The element must contain one `radius-shared-secret` child element.

**radius-shared-secret**

This element defines the RADIUS shared secret file.

The path to the secret file can be given as a value of the `file` attribute.

Alternatively, the secret can be included as the contents of the `radius-shared-secret` element.

**submethod-generic**

This element sets the generic submethod in use. This element can be used to add custom submethods to keyboard-interactive authentication.

The `name` on the method must be given in the attribute.

Optional `params` for the submethod can be given as well.

**auth-gssapi**

This element sets the GSSAPI authentication method.

The `dll-path` can be given as an attribute. This specifies where the necessary GSSAPI libraries are located. If this attribute is not specified, the libraries are searched for in a number of common locations. The full path to the libraries should be given, for example, `"/usr/src/krb5-1.3.5/src/lib/libkrb5.so,/usr/src/krb5-1.3.5/src/lib/libgssapi_krb5.so"`.

The `allow-ticket-forwarding` attribute defines whether the server allows forwarding the Kerberos ticket over several connections. The attribute can have a value of `yes` or `no`. The default is `no`.

**authentication**

The `authentication` elements can be nested within each other. The child method(s) must be passed in addition to the parent method.

A sample `authentication` element is shown below:

```
<authentication action="deny">
  <auth-publickey />
  <authentication action="allow" set-group="local-user">
    <selector>
      <ip address="10.1.55.14-10.1.55.99" />
      <user name="johnd" />
      <user name="janed" />
    </selector>
    <auth-keyboard-interactive max-tries="4"/>
      <submethod-radius>
        <radius-server address="10.1.61.128">
          <radius-shared-secret file="&configdir;/radius-secret-file"/>
        </radius-server>
      </submethod-radius>
    </auth-keyboard-interactive>
  </authentication>
  <authentication action="allow" set-group="finance-inspector">
    <selector>
      <user-group name="finance" />
    </selector>
    <auth-password />
  </authentication>
</authentication>
```

In the example above, all users are first required to authenticate using public-key authentication. Based on selector matching, also a second method needs to be passed (RADIUS via keyboard-interactive or password). A group is set based on the matched and passed authentication methods. If a user does not match to either of the child authentication elements, access is denied (the parent authentication element has the action set to `deny`).

A sample `authentication` element that sets requirements for certificate authentication is shown below:

```
<authentication action="allow">
  <auth-publickey />
  <authentication action="allow" set-group="admin">
    <selector>
      <user-privileged value="yes" />
      <certificate field="ca-list" pattern="exa-ca1,exa-ca2" />
      <certificate field="subject-name" pattern="C=FI, O=SSH, CN=%username%" />
      <certificate field="altname-email" pattern="%username%@ssh.com" />
      <certificate field="altname-upn" pattern="%username%@ssh" />
    </selector>
  </authentication>
  <authentication action="allow">
    <selector>
      <publickey-passed length="1024-2048" />
    </selector>
  </authentication>
  <authentication action="deny" />
</authentication>
```

In the example above, privileged users (administrators) are required to pass certificate authentication and the certificate must contain the correct fields. Other users are allowed to log in using a plain public key of a size from 1024 to 2048 bits.

Specifying an explicit `deny` action last is necessary in a restrictive policy, as otherwise a non-matching connection would use the implicit default `allow` rule, which contains the default authentication methods. Alternatively, the action of the parent authentication element can be set to `deny`.

## The `services` Element

The `services` element defines the policy for the various services the server offers.

The services element contains one or more rules (`rule`) and optionally defines groups (`group`).

**group**

Creates a group that can be used a basis for restricting services. Groups are defined based on selectors.

If the user was already put to a group during authentication using the `set-group` attribute, the group definitions in the `services` element are ignored.

**selector**

The selectors define the users that belong to the group. The same selectors can be used as in the `authentication-methods` element. See the section called “Selectors” and the section called “The `authentication-methods` Element”.

A sample `group` element is shown below:

```
<!-- Partners. -->
<group name="partner">
  <selector>
    <ip address="172.16.0.0/12" />
    <user name="sjx" id="24814" />
    <user name="mtx" id="17692" />
  </selector>
  <selector>
    <certificate field="issuer-name" pattern="C=FI, O=Friend Oyj, CN=FriendCA" />
  </selector>
</group>

<!-- Remote access. -->
<group name="remote-access">
  <selector>
    <interface address="63.81.17.62" />
  </selector>
</group>

<!-- Password change needed. -->
<group name="passwd-change">
  <selector>
    <user-password-change-needed/>
  </selector>
</group>

<!-- Backup. -->
<group name="backup">
  <selector>
    <user name="backup" />
  </selector>
</group>
```

**rule**

This element defines a rule for the specified `group` of users. Rules can be used to restrict the services and commands the server allows to the users. The element can have three attributes: `group`, `idle-timeout`, and `print-motd`.

The rules are read in order, and the first rule that matches the user's `group` is used. The match must be exact. No wild cards are allowed in the `group` attribute. If no `group` is specified, the rule matches to all users.

The `idle-timeout` attribute sets the idle timeout limit in seconds. If the connection has been idle (all channels) this long, the connection is closed. Default is `0` (zero), which disables idle timeouts.

The `print-motd` attribute defines whether the message of the day (`/etc/motd`) is printed when a user logs in interactively to a Unix server. The value must be `yes` or `no`. The default is `yes`.

Each rule can contain `environment`, `terminal`, `subsystem`, `tunnel-agent`, `tunnel-x11`, `tunnel-local`, `tunnel-remote`, and `command` elements.

An empty rule allows the specified group to perform all actions.

A default unnamed rule allows all users access to all services. The unnamed rule should be kept as the last rule, so it will match to users that are not set in any group, but you should edit the rule according to your security policy.

**environment**

This element defines the environment variables the user group can set. The variables are given as a value of the `allowed` attribute. By default, the user can set the `TERM`, `PATH`, `TZ`, `LANG`, and `LC_*` variables.

Allowed variables are normally matched case-insensitively. Alternatively, the variables can be specified using the `allowed-case-sensitive` attribute.

**terminal**

This element defines whether terminal access is allowed or denied for the user group. The word `allow` or `deny` can be given as the value of the `action` attribute. By default, terminal access is allowed.

On Unix systems, the `chroot` attribute can be optionally used to define a directory where the user is chrooted during the terminal session.

If terminal access is denied, also shell commands are denied, unless the commands are specifically allowed by the `command` element.

**subsystem**

This element defines a subsystem. The subsystem `type` must be given as an attribute. The use of the subsystem can be allowed or denied by giving the word `allow` or `deny` as the value of the `action` attribute. The default action is `allow`.

Optionally also the `application` can be given. This is not necessary with the SFTP subsystem if the SFTP binary is in the default location.

On Unix systems, the `chroot` attribute can be optionally used to define a directory where the user is chrooted when running the subsystem.

The element can contain multiple `attribute` child elements.

**attribute**

This element defines an attribute of a subsystem.

The `name` must be given as an attribute. The `value` can be optionally given.

This can be used, for example, on Windows platforms to set the user home directory and virtual folders for SFTP.

```
<subsystem type="sftp" application="sft-server-g3.exe">
  <attribute name="home" value="%USERPROFILE%" />
  <attribute name="virtual-folder" value="c=c:\" />
</subsystem>
```

**command**

This element defines a shell command as allowed, denied, or forced. The element can have four attributes, however, all of them are not used at the same time: `action`, `application`, `application-case-sensitive`, and `chroot`.

The value of the `action` attribute can be either `allow`, `deny`, or `forced`. The default is `allow`.

If the `deny` action is set, all shell commands are denied and no further attributes should be specified.

For the `allow` and `forced` actions, the `application` must be given as an attribute. If the application is not given for the `allow` action, all commands are allowed.

If the `forced` action is set, the specified application is run automatically when the user logs in.

Applications are normally matched case-insensitively. Alternatively, the application can be specified using the `application-case-sensitive` attribute.

On Unix systems, the `chroot` attribute can be optionally used to define a directory where the user is chrooted when running the command.

**tunnel-agent**

This element defines whether agent tunneling (forwarding) is allowed or denied by the server.

The word `allow` or `deny` can be given as the value of the `action` attribute. By default, agent forwarding is allowed.

For more information, see Section 8.5.

**tunnel-x11**

This element defines whether X11 tunneling (forwarding) is allowed or denied by the server.

The word `allow` or `deny` can be given as the value of the `action` attribute. By default, X11 forwarding is allowed.

For more information, see Section 8.4.

**tunnel-local**

This element defines a rule for local TCP tunnels (port forwarding).

The word `allow` or `deny` can be given as the value of the `action` attribute. By default, local tunnels are allowed.

Tunneling settings can be further defined with the `src` and `dst` elements.

For more information on tunneling, see Section 8.2.

**src**

This element defines a source address for local and remote TCP tunnels.

The `address` or the `fqdn` (not both) can be given as an attribute. Also the `port` can be given.

**dst**

This element defines a destination address for local TCP tunnels.

The `address` or the `fqdn` (not both) can be given as an attribute. Also the `port` can be given.

**tunnel-remote**

This element defines a rule for remote TCP tunnels (port forwarding).

The word `allow` or `deny` can be given as the value of the `action` attribute. By default, remote tunnels are allowed.

Tunneling settings can be further defined with the `src` and `listen` elements.

For more information on tunneling, see Section 8.3.

**src**

This element defines a source address for local and remote TCP tunnels.

The `address` or the `fqdn` (not both) can be given as an attribute. Also the `port` can be given.

**listen**

This element defines a listen address for remote TCP tunnels.

The `address` and the `port` can be given as attributes.

A sample `rule` element is shown below:

```
<!-- The finance inspector. -->
<rule group="finance-inspector" print-motd="no">
  <tunnel-local action="allow">
    <!-- Microsoft SQL ports. -->
    <dst fqdn="finance-db.example.com" port="1433" />
    <dst fqdn="finance-db.example.com" port="1434" />
  </tunnel-local>
  <!-- Can execute commands and shells, as no overriding behavior
       is defined. -->
</rule>

<!-- Administrators are allowed to do anything. -->
<rule group="admin" />

<!-- Remote access. -->
<rule group="remote-access" idle-timeout="600">
```

```
  <!-- Setting terminal action to "deny" also denies shell
       commands, unless they are specifically allowed.     -->
  <terminal action="deny" />
  <subsystem type="sftp" application="sft-server-g3" chroot="%homedir%" />
  <tunnel-local action="allow">
    <!-- IMAP. -->
    <dst fqdn="imap.example.com" port="143" />
    <dst fqdn="imap.example.com" port="993" />
    <!-- POP. -->
    <dst fqdn="mail.example.com" port="109" />
    <dst fqdn="mail.example.com" port="110" />
    <dst fqdn="mail.example.com" port="995" />
  </tunnel-local>
</rule>

<rule group="backup">
  <terminal action="deny" />
  <!-- This account is only used to back up the disk drive. -->
  <command application="dd if=/dev/hda" action="forced" />
  <tunnel-local action="deny" />
  <tunnel-remote action="deny" />
</rule>

<!-- This rule is used to force password change. -->
<rule group="passwd-change">
  <terminal action="deny"/>
  <command application="/usr/bin/passwd" action="forced" />
  <tunnel-local action="deny" />
  <tunnel-remote action="deny" />
</rule>

<!-- Default miscellaneous rule. -->
<rule>
  <!-- All others will be denied. There is no "denied" setting. -->
  <environment allowed="TERM,PATH,TZ,LANG,LC_*" />
  <terminal action="deny" />
  <subsystem type="sftp"
             application="sft-server-g3"/>
  <!-- Other commands will be denied. -->
  <command application="date" action="allow" />

  <!-- Other local tunnels will be denied. -->
  <tunnel-local action="allow">
    <dst fqdn="imap.example.com" port="143" />
    <dst fqdn="imap.example.com" port="993" />
    <dst fqdn="*.example.com" port="260" />
  </tunnel-local>

  <!-- Other remote tunnels will be denied. -->
  <tunnel-remote action="allow">
    <listen port="6881" />
    <src fqdn="*.example.com" port="220" />
  </tunnel-remote>

</rule>
```

## Authors

SSH Communications Security Corp.

For more information, see http://www.ssh.com.

## See Also

ssh-server-g3(8), ssh-server-config-tool(8), ssh-broker-config(5)

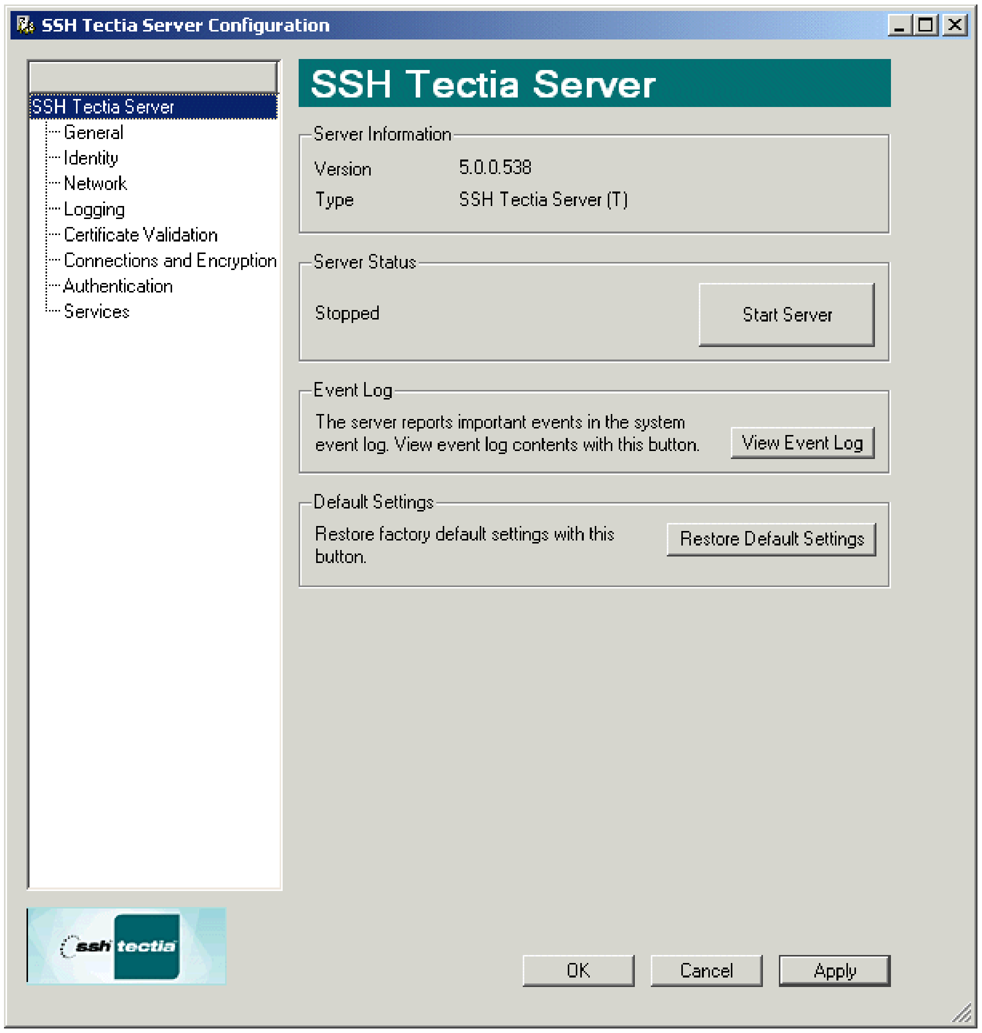
## 4.1 Configuration Tool (Windows)

On Windows, the easiest way to configure the server is to use the **SSH Tectia Server Configuration** tool. Start the program by clicking the **SSH Tectia Server Configuration** icon in the **SSH Tectia Server** program group (or by running the `ssh-server-gui.exe` program located in the installation directory).

The **SSH Tectia Server Configuration** tool displays the settings in a tree structure. Select the desired configuration page by clicking the list displayed on the left.

### 4.1.1 SSH Tectia Server

The **SSH Tectia Server** page of the **Configuration** tool allows you to start and stop the server, view the event log and restore the settings to their default values.



SSH Tectia Server

Server Information

Version 5.0.0.538

Type SSH Tectia Server (T)

Server Status

Stopped

Event Log

The server reports important events in the system event log. View event log contents with this button.

Default Settings

Restore factory default settings with this button.

**Figure 4.1. SSH Tectia Server Configuration - SSH Tectia Server page**

**Server Information**

The server version, release date, and server type (A, F, or T) are shown at the top of the page.

**Server Status**

The server status is displayed on the left. The status can be either **Stopped**, **Starting**, **Running**, **Stopping**, or **Failure**. To start or stop the server, click the **Start Server/Stop Server** button.

**Event Log**

Important events are logged in the system event log. Click the **View Event Log** button to launch the **Event Viewer** program that allows you to examine the log contents. See Section 6.2 for more information.

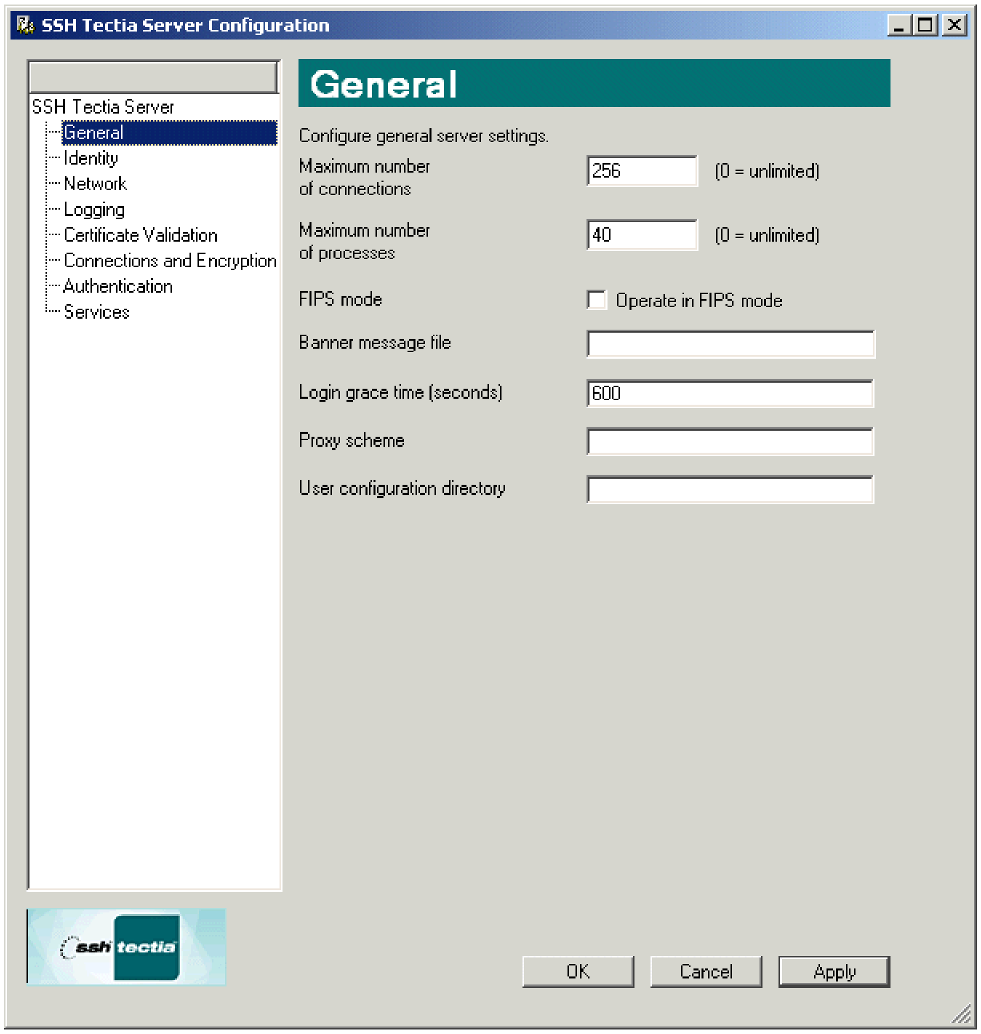
**Default Settings**

To discard any changes you have made to the configuration settings and restore the factory default values, click the **Restore Default Settings** button and click **OK** in the confirmation dialog.

Note that the settings will not revert back to the previously saved values, but to the initial values that are built into the program.

## 4.1.2 General

The **General** page of the **SSH Tectia Server Configuration** tool contains the general server settings, for example, the maximum number of connections, FIPS mode, and banner message.



General

Configure general server settings.

| | | |
|---|---|---|
| Maximum number of connections | 256 | (0 = unlimited) |
| Maximum number of processes | 40 | (0 = unlimited) |
| FIPS mode | ☐ Operate in FIPS mode | |
| Banner message file | | |
| Login grace time (seconds) | 600 | |
| Proxy scheme | | |
| User configuration directory | | |

**Figure 4.2. SSH Tectia Server Configuration - General page**

**Maximum number of connections / Maximum number of processes**

Specify how many connections and processes at most the server program will handle simultaneously. SSH Tectia Server uses a distributed architecture where the master server process launches several servant server processes that handle the actual connections.

**Maximum number of processes** defines the maximum number of servant processes the server will launch. **Maximum number of connections** defines the maximum number of client connections allowed by each servant.

If you specify `0` (zero) as the value, the number of connections (or processes) uses the default values for the operating system.

Limiting the maximum number of connections is useful in systems where a high load in the server program when opening new connections may cause system overload.

**FIPS mode**

SSH Tectia Server can be operated in *FIPS mode*, using a version of the cryptographic library that has been validated according to the Federal Information Processing Standard (FIPS) 140-2. In this mode, the cryptographic operations are performed according to the rules of the FIPS 140-2 standard.

The software uses standard libraries by default - the FIPS 140-2 validated libraries are available separately. If the FIPS-certified cryptographic library has been installed, SSH Tectia Server will detect and use it automatically.

For a list of platforms on which the FIPS library has been validated or tested, see *SSH Tectia Client/Server Product Description*.

Select the **Enable FIPS Mode** check box to use the FIPS-certified version of the SSH cryptographic library.

**Note**

The system does not actually check whether the FIPS-certified version of the library has been installed.

**Banner message file**

Specify the path to the message that is sent to the client before authentication. Note, however, that the client is not obliged to show this message.

**Login grace time**

Specify a time after which the server disconnects if the user has not successfully logged in. If the value is set to `0`, there is no time limit. The default is `600` (seconds).

**Proxy scheme**

Define rules for HTTP or SOCKS proxy servers that SSH Tectia Server will use when a client forwards a connection (local tunnel).

For a format of the proxy scheme, see the description of the `settings` element in ssh-server-config(5).

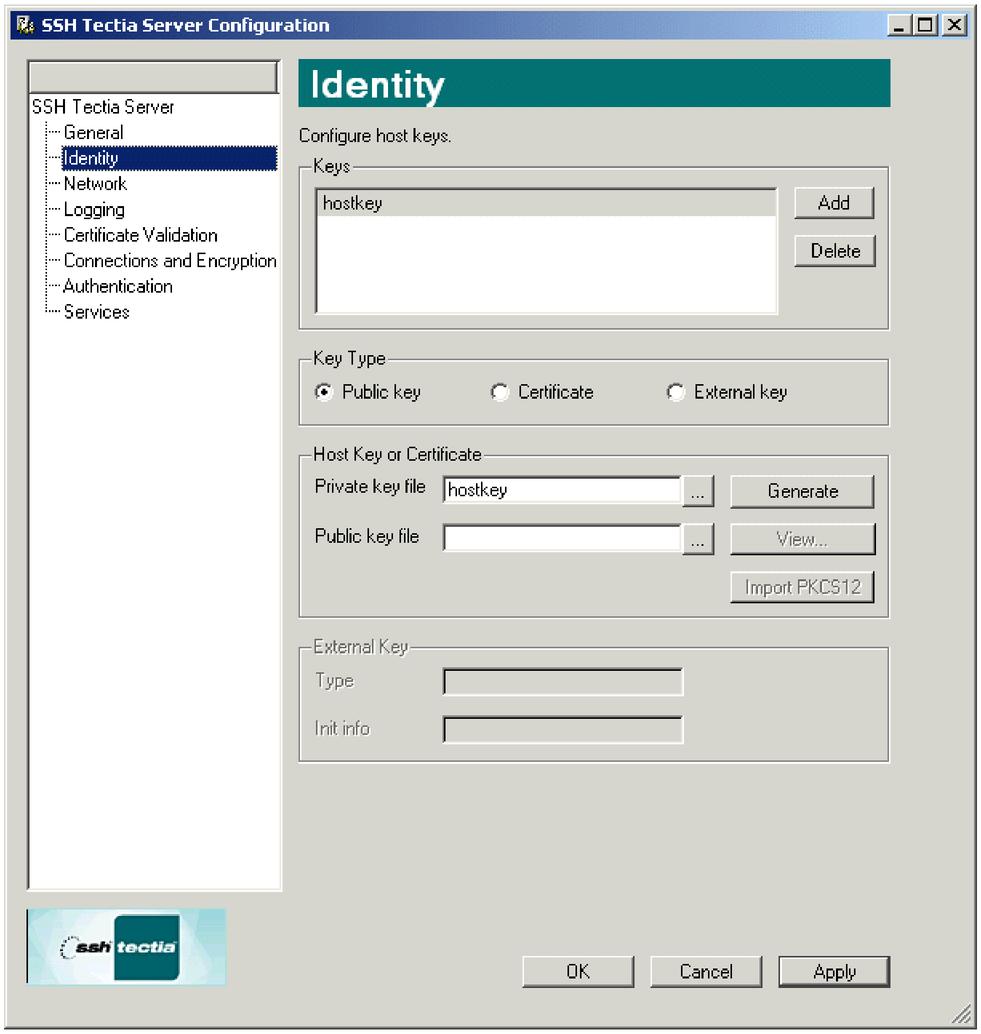
**User configuration directory**

Specify a path to user-specific configuration data. With this, the administrator can control those options that are usually controlled by the user. This is given as a pattern string which is expanded by SSH Tectia Server. In the string, `%D` is the user's home directory, `%U` is the user's login name, `%IU` is the user's user ID (uid), and `%IG` is user's group ID (gid).

The server has to be restarted to use the changed setting.

## 4.1.3 Identity

The **Identity** page of the **SSH Tectia Server Configuration** tool is used to specify the host keys and host certificates that identify the server to the clients.



Configure host keys.

Keys: hostkey

Key Type: [✓ Public key] Certificate External key

Host Key or Certificate

Private key file: hostkey

Public key file:

External Key

Type:

Init info:

Figure 4.3. SSH Tectia Server Configuration - Identity page

**Keys**

To add a host key pair, click **Add**.

To delete a host key pair, select a key pair from the list and click **Delete**.

**Key Type**

Select a key type for the key pair. This can be plain **Public key**, **Certificate**, or **External key**.

**Host Key or Certificate**

This setting is different depending on whether you have selected **Public key** or **Certificate** as the key type.

**Private key file**

Click the ellipsis (**...**) button on the right-hand side of the text field to change the private host key file. The **Select File** dialog appears, allowing you to specify the desired file. You can also type the path and filename directly in the text field.

The default file is `hostkey`, located in the installation directory (by default, `C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Server`). The key file should be readable only by the server itself.

Click the **Generate** button to generate a new host key pair. This launches the `ssh-keygen-g3.exe` command-line tool and generates a 2048-bit DSA key pair. If you want to generate some other type of key, see Appendix ssh-keygen-g3(1) for instructions on how to manually generate the key pair.

**Public key file**

This setting is active if you have selected plain **Public key** as the key type.

Click the ellipsis button (**...**) on the right-hand side of the text field to change the public host key file. The **Select File** dialog appears, allowing you to specify the desired file. You can also type the path and filename directly in the text field.

If the public key is not specified, it will be derived from the private key. However, specifying the public key will decrease the start-up time for the software, as deriving the public key is a fairly slow operation.

**Certificate file**

This setting is active if you have selected **Certificate** as the key type.

Click the ellipsis (**...**) button to select the host certificate file. The **Select File** dialog appears, allowing you to specify the desired file. You can also type the path and file name directly in the text field.

Click the **View** button to display the current certificate.

Click the **Import** button to import a private key stored in the Personal Information Exchange (PFX) format. The **Select File** dialog appears, allowing you to specify the desired file.

**External Key**

This setting is active only if you have selected **External key** as the key type.

Enter the **Type** of the external key (for example, `entrust`) in the text box.

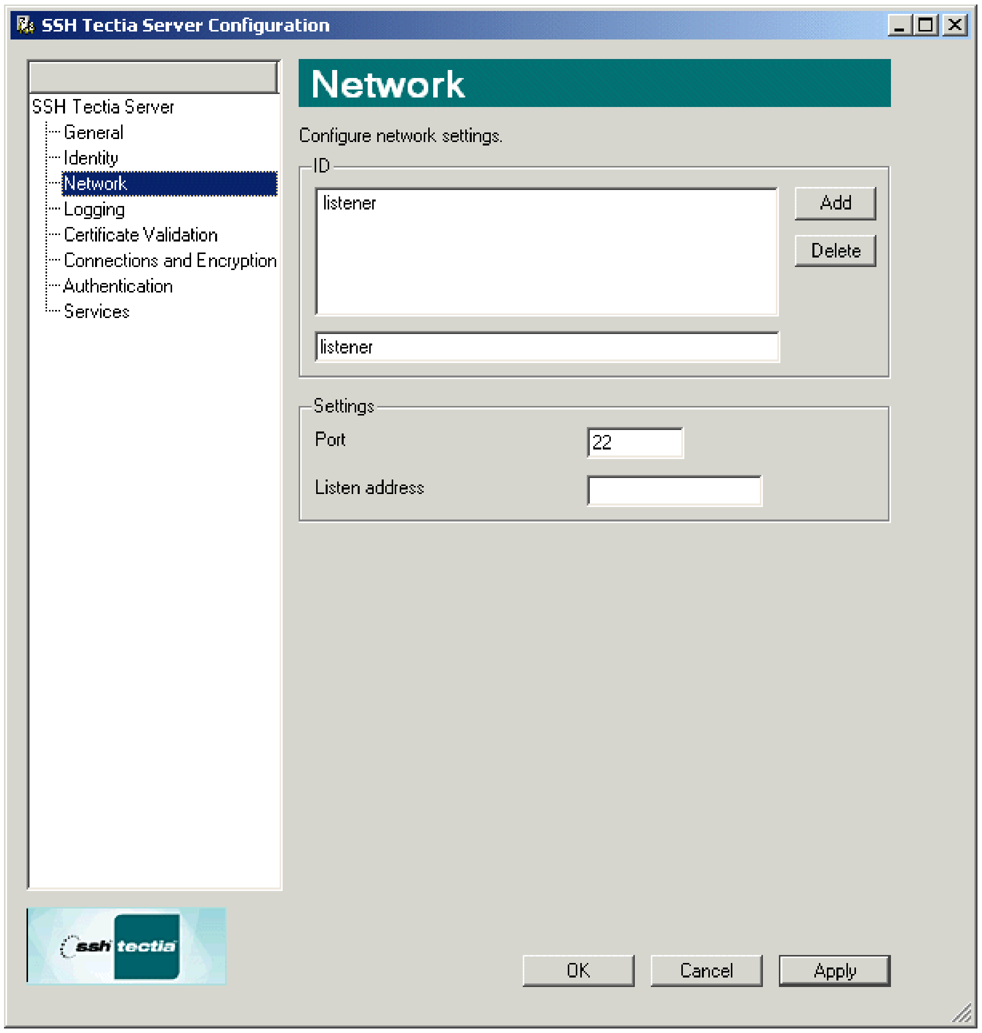
Enter the **Initialization info** of the external key.

Please note that all key and certificate files should be located on a local drive. Network or mapped drives should not be used, as the server program may not have proper access rights for them.

See also Section 5.1, Section 5.2, and Section 5.3.

### 4.1.4 Network

The **Network** page of the **SSH Tectia Server Configuration** tool allows you to specify the basic network settings to be used.



Network

Configure network settings.

ID

listener

listener

Settings

Port 22

Listen address

**Figure 4.4. SSH Tectia Server Configuration - Network page**

**ID**

The **ID** list shows the interfaces SSH Tectia Server is listening to. You can specify several listeners to different addresses. Also multiple ports at the same address can be listened to.

To add a new network listener, click the **Add** button and give the ID in the text box below. The ID must be unique. Give also the **Port**, and **Listen address** (see below).

To remove a listener, select an ID from the list and click **Delete**.

**Port**

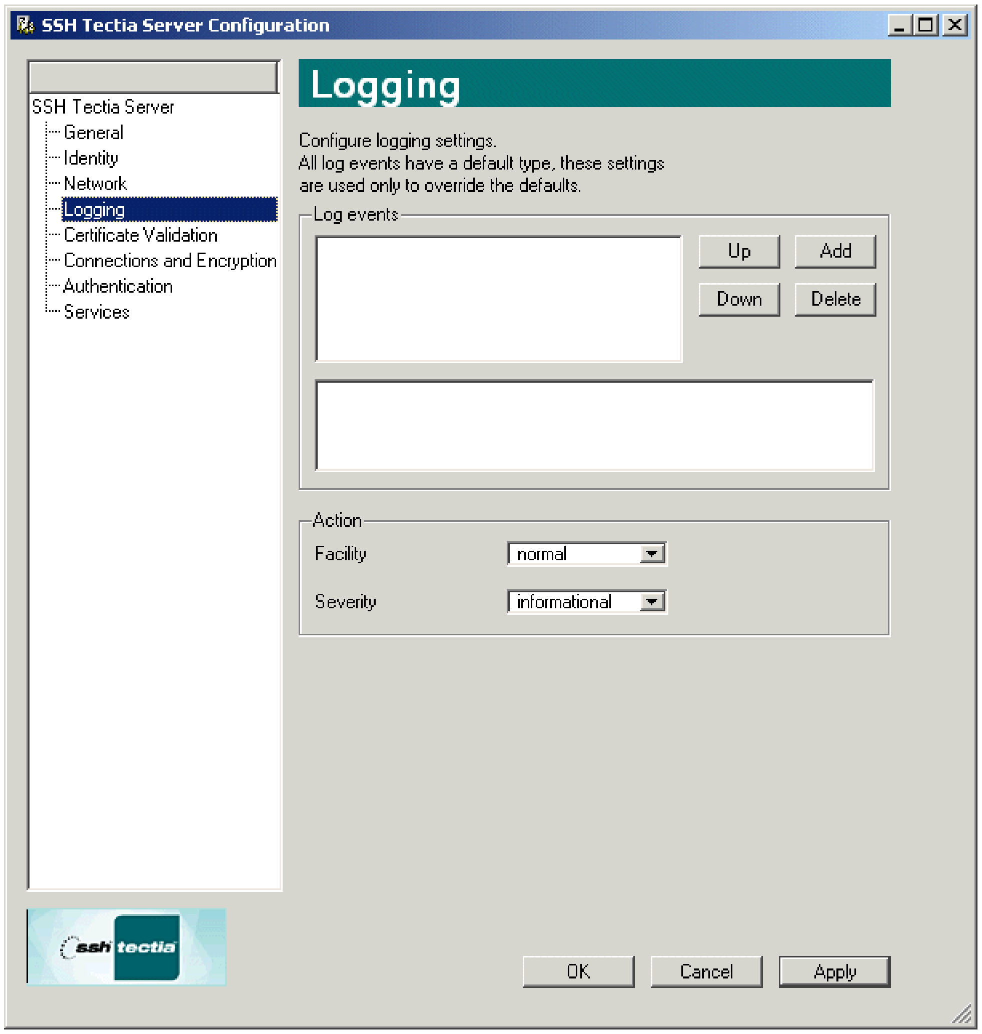
Specify the port number that the server listens on (allowed values are 1 - 65535). The server has to be restarted in order to use the changed setting. The default port is 22.

**Listen address**

Specify the IP address of the network interface card where the Secure Shell server socket is bound. The server has to be restarted to use the changed setting.

## 4.1.5 Logging

The **Logging** page of the **SSH Tectia Server Configuration** tool allows you to specify what kind of information is logged in the event log.



Configure logging settings.
All log events have a default type, these settings
are used only to override the defaults.

Facility: normal

Severity: informational

**Figure 4.5. SSH Tectia Server Configuration - Logging page**

You can set the type of different logging events. The events have reasonable default values, which are used if no explicit logging settings are made. This page allows customizing the default values.

Click **Add** to add a new logging rule. Enter the name of the log event in the text box and select the **Facility** and **Severity** from the drop-down list. See Appendix D for a full list of the log event names.

On Windows, only the **Normal** and **Discard** facilities are used. Setting the facility to **Discard** causes the server to ignore the specified log events.

On Windows, the following event severities are used:

- **Information**
- **Warning**
- **Error**
- **Security success**
- **Security failure**
- **Ignore** (The event is not logged.)

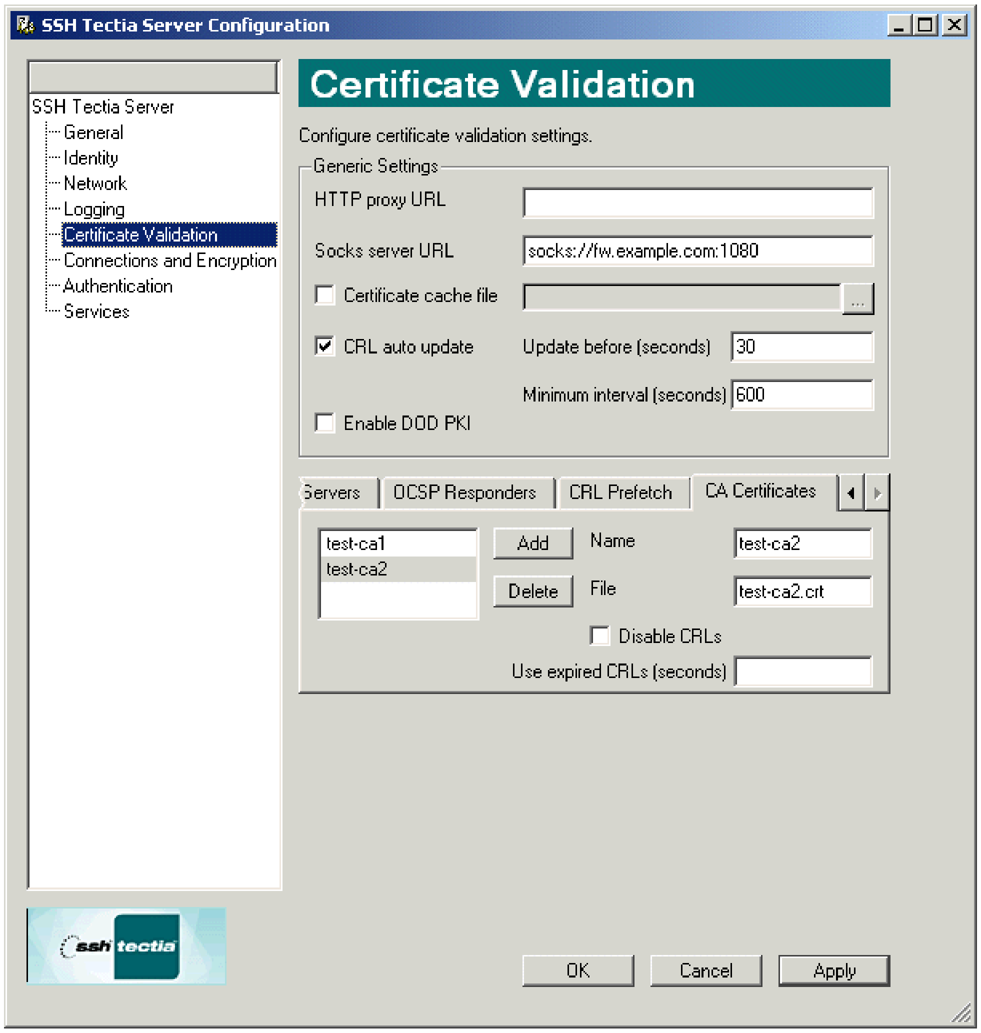
See Section 6.2 for more information on the event types.

To change the order of the logging rules, select a rule from the list, and click **Up** and **Down** to move the log event. If several rules match to the same log event, the first matching rule on the list is used.

Click **Delete** to delete a logging rule.

### 4.1.6 Certificate Validation

On the **Certificate Validation** page you can configure certification authorities (CA) that are trusted in user authentication.



Certificate Validation

Configure certificate validation settings.

Generic Settings

HTTP proxy URL

Socks server URL: socks://fw.example.com:1080

☐ Certificate cache file

☑ CRL auto update — Update before (seconds): 30

Minimum interval (seconds): 600

☐ Enable DOD PKI

test-ca1
test-ca2

Name: test-ca2

File: test-ca2.crt

☐ Disable CRLs

Use expired CRLs (seconds)

**Figure 4.6. SSH Tectia Server Configuration - Certificate Validation page**

## Generic Settings

Generic settings apply to all CA certificates and CRL fetching.

### HTTP proxy URL

Define a HTTP proxy address if such is required for making LDAP or OCSP queries for certificate validity.

### SOCKS server URL

Define a SOCKS server address if such is required for making LDAP or OCSP queries for certificate validity.

### Certificate cache file

Select the check box to enable certificate caching.

Click the ellipsis (**...**) button to select the cache file where the certificates and CRLs are stored when the SSH Tectia Server service is stopped, and read back in when the service is restarted. The **Select File** dialog appears, allowing you to specify the desired file. You can also type the path and file name directly in the text field.

**CRL auto update**

Select the check box to enable CRL auto update.

When auto update is on, SSH Tectia Server periodically tries to download the new CRL before the old one has expired. The **Update before** field specifies how many seconds before the expiration the update takes place. The **Minimum interval** field sets a limit for the maximum update frequency.

**Enable DOD PKI**

Select this check box if the certificates are required to be compliant with the DoD PKI (US Department of Defense Public-Key Infrastructure).

**LDAP Servers**

On the **LDAP Servers** tab, you can define LDAP servers that are used for fetching certificate revocation lists (CRLs).

If a CRL distribution point is defined in the certificate, the CRL is automatically retrieved from that address. Otherwise, the LDAP servers specified here are used.

To add an LDAP server, click **Add** and enter the **Address** and **Port** of the server. The default port is `389`

To delete an LDAP server, select a server from the list and click **Delete**.

**OCSP Responders**

On the **OCSP Responders** tab, you can define OCSP responder servers that are used for Online Certificate Status Protocol queries.

If the certificate has a valid `Authority Info Access` extension with an OCSP Responder URL, it is used instead of this setting. Note that for the OCSP validation to succeed, both the end-entity certificate and the OCSP Responder certificate must be issued by the same CA.

To add an OCSP responder, click **Add** and enter the **URL** of the server. Optionally, you can also enter a **Validity period** in seconds for the OCSP data. During this time, new OCSP queries for the same certificate are not made but the old result is used.

To delete an OCSP responder, select a responder from the list and click **Delete**.

**CRL Prefetch**

On the **CRL Prefetch** tab, you can define addresses from which CRLs are periodically downloaded.

To add a CRL prefetch address, click **Add** and enter the **URL** of the CRL distribution point and the **Interval** how often the CRL is downloaded. The default interval is `3600` (seconds).

To delete CRL prefetch address, select an address from the list and click **Delete**.

**CA Certificates**

On the **CA Certificates** tab, you can define the CA certificates that are trusted for user authentication.

To add a CA certificate as trusted, click **Add** and enter the **Name** of the CA and the location of the certificate **File**.

The CA **Name** can be referred to in the selectors on the Authentication page. See Section 4.1.9.

**Disable CRLs**

Select the check box to stop using the certificate revocation list. This option is should be used for testing purposes only!

**Use expired CRLs**

Specify in seconds how long expired CRLs are used.

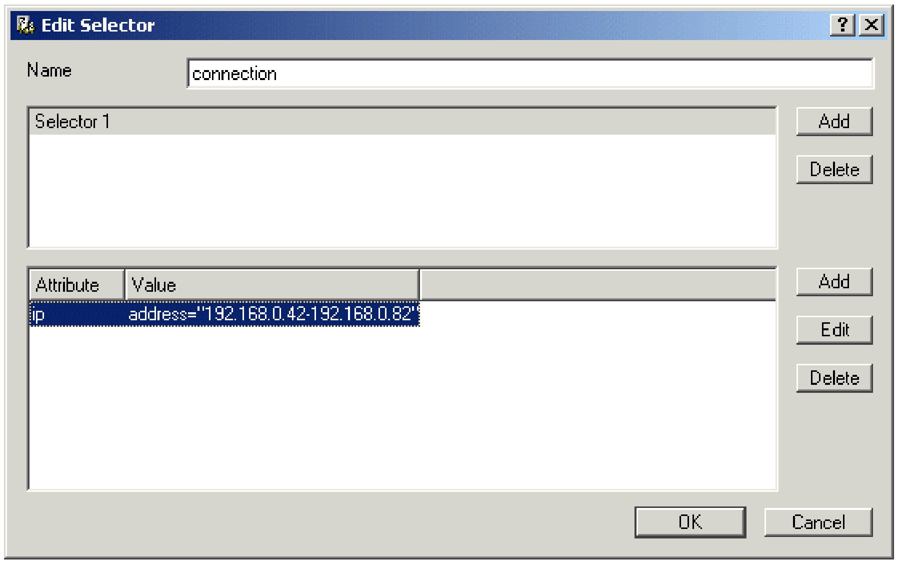
To remove a CA from the trusted CAs, select a CA from the list and click **Delete**.

### 4.1.7 Editing Selectors

On the **Connections and Encryption**, **Authentication**, and **Services** pages, different *selectors* can be used to set access rules to users based on the user parameters such as username or location. Users can be divided to groups dynamically, for example, based on the authentication method they used for logging in. On the **Services** page, each group can then be allowed or denied services such as tunneling, file transfer, and terminal access.

See the section called “Selectors” for more information on selector handling rules.

The **Edit Selector** dialog box is opened from the **Connections and Encryption**, **Authentication**, and **Services** pages whenever you create new items that can include selectors. See Figure 4.7.



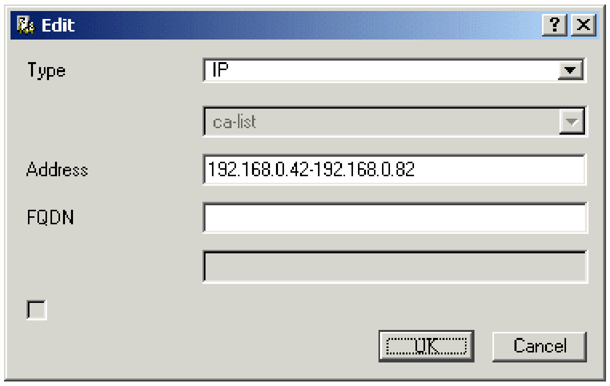
Name: connection

| Attribute | Value |
|---|---|
| ip | address="192.168.0.42-192.168.0.82" |

**Figure 4.7. The Edit Selector dialog box**

To add a selector to the element, click the topmost **Add** button.

The selector is initially empty. To add items to the selector, choose a selector from the list and click the lowermost **Add** button. The **Edit** dialog box opens (Figure 4.8).



Type: IP

Address: 192.168.0.42-192.168.0.82

**Figure 4.8. Editing a selector**

**Edit**

Select the **Type** of the selector item from the drop-down list. The attributes of the selector depend on the type.

**Interface**

The **Interface** selector matches to the listener interface **ID** or **Address** and/or **Port**. At least one attribute must be given. If the **ID** is defined, the others MUST NOT be given. If the **ID** is not defined, either or both of **Address** and **Port** may be given.

**IP**

The **IP** selector matches to an IP **Address** or **FQDN** (fully qualified domain name) of the client. Either **Address** or **FQDN** can be given, not both.

The **Address** can be in one of the following formats:

- an IP address range of the form `x.x.x.x-y.y.y.y`
- an IP mask of the form `x.x.x.x/y`
- a single IP address `x.x.x.x`

The **FQDN** attribute matches to an FQDN pattern (case-insensitive). The form of the pattern is not checked.

**Certificate**

This selector matches to a **Pattern** in a specified **Field** of the user certificate.

The field can be either **ca-list**, **issuer-name**, **subject-name**, **serial-number**, **altname-email**, **altname-upn**, **altname-ip**, or **altname-fqdn**.

The format of the pattern depends on the type of the field. The **ca-list** field contains a list of CA names separated by commas. The names that are defined in the **ca-certificate** element are used. The **issuer-name** and **subject-name** fields contain distinguished names, **serial-number** a positive integer. The **altname-fqdn** field contains a hostname and **altname-ip** an IP address or a range. The **altname-email** field contains an email address and **altname-upn** the principal name.

The **altname-fqdn**, **altname-upn**, **altname-email**, **subject-name**, and **issuer-name** selectors may contain the **%username%** keyword which is replaced with the user's name before comparing with the actual certificate data. The **%hostname%** keyword can be used in the same way and it is replaced by the client's FQDN. These patterns may also contain "*" and "?" globbing characters.

Patterns are normally matched case-insensitively. Alternatively, the pattern can be specified using the **Pattern case-sensitive** attribute.

**Host certificate**

This selector matches to a **Pattern** in a specified **Field** of the client host certificate.

The field can be either **ca-list**, **issuer-name**, **subject-name**, **serial-number**, **altname-email**, **altname-upn**, **altname-ip**, or **altname-fqdn**.

Patterns are normally matched case-insensitively. Alternatively, the pattern can be specified using the **Pattern case-sensitive** attribute.

**User**

This selector matches to a user **Name** or **ID**. A list of usernames or IDs can be given as a comma-separated list.

Names are normally matched case-insensitively. Alternatively, the names can be specified using the **Name case-sensitive** attribute.

**User group**

This selector matches to a user group **Name** or **ID**. A list of user-group names or IDs can be given as a comma-separated list.

Names are normally matched case-insensitively. Alternatively, the names can be specified using the **Name case-sensitive** attribute.

**User privileged**

This selector matches to a privileged user (administrator or root) or to a non-privileged user. Select the **Value** check box to match to a privileged user or clear it to match to a normal user.

**Blackboard**

This selector matches to a **Pattern** in a specified blackboard **Field**.

Patterns are normally matched case-insensitively. Alternatively, the pattern can be specified using the **Pattern case-sensitive** attribute.

**Public key passed**

This selector matches if authentication is passed using a normal public key (without a certificate). Optionally, the **Length** range of the public key can be given as an attribute, for example **1024-2048**.

Click **OK** to create the selector item.

To edit selector attributes, choose a selector item from the lowermost list and click **Edit**. The **Edit Selector** dialog box opens (Figure 4.8).

To remove a selector item, choose a selector item from the list and click the lowermost **Delete** button.

To remove a selector, choose a selector from the topmost list and click the topmost **Delete** button.

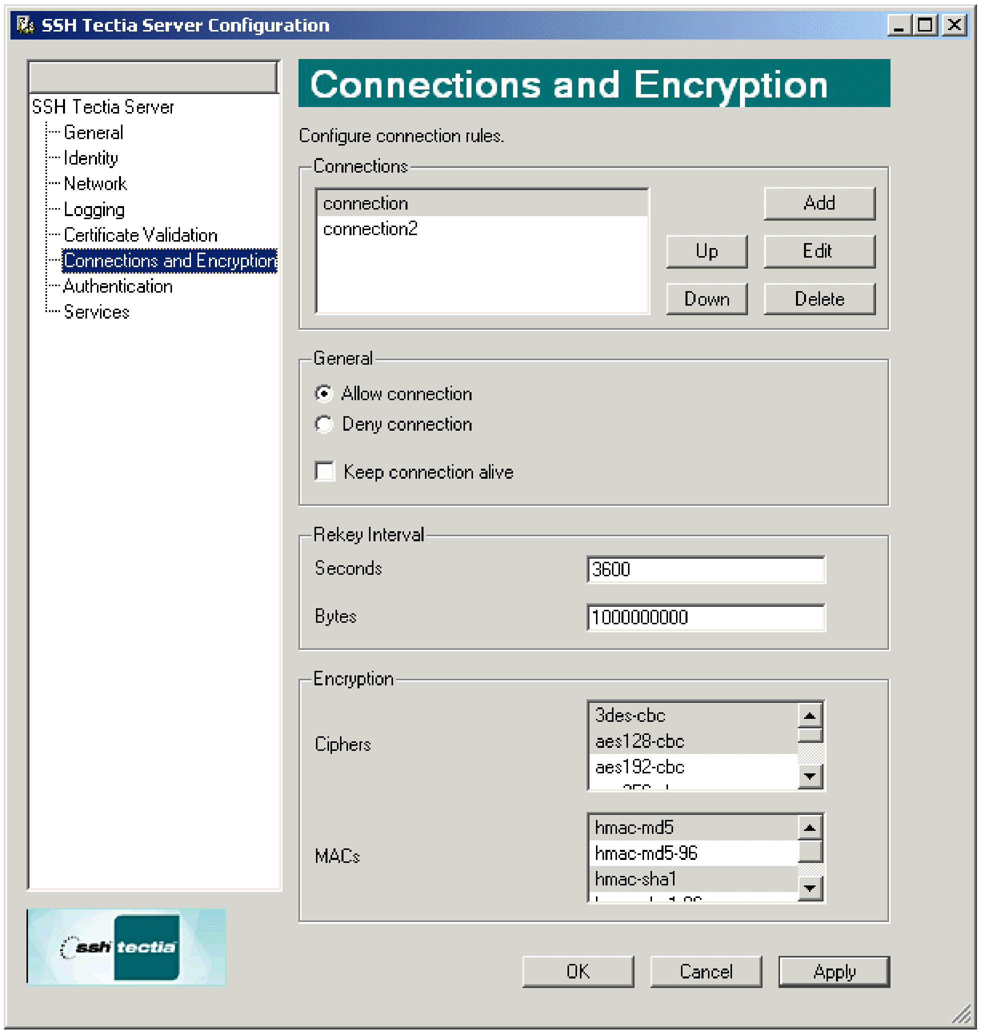
Click **OK** to the accept the changes and return to the main **SSH Tectia Server Configuration** page.

## 4.1.8 Connections and Encryption

On the **Connections and Encryption** page you can create connection rules that restrict connections based on various selectors and set ciphers and MACs used for the connections.

The selectors define which connections a connection rule applies to. The order of the rules is important. The first matching rule is used and the remaining rules are ignored.

If no selectors (or only empty selectors) are specified in a connection rule, the rule matches to all connections.



Configure connection rules.

Connections: connection, connection2

General: (● Allow connection) Deny connection; ☐ Keep connection alive

Rekey Interval: Seconds 3600; Bytes 1000000000

Encryption: Ciphers 3des-cbc, aes128-cbc, aes192-cbc; MACs hmac-md5, hmac-md5-96, hmac-sha1

Figure 4.9. SSH Tectia Server Configuration - Connections and Encryption page

**Connections**

To add a new connection rule, click **Add** next to the **Connections** list. The **Selector** dialog box opens. See Section 4.1.7 for information on editing the selectors.

Only the **Interface** and **IP** selectors are relevant for connection rules. For example, the username is not yet available when the connection rules are processed. See the section called “Selectors”.

To edit a connection rule, select a connection from the list and click **Edit**. The **Selector** dialog box opens. See Section 4.1.7.

To change the order of the connection rules, choose a connection from the list and click the **Up** and **Down** buttons. The rules are read in order, and the first matching connection rule on the list is used.

To remove a connection rule, select a rule from the list and click **Delete**.

**General**

Select whether the connection is allowed or denied.

**Keep connection alive**

Select this check box to send keepalive messages to the other side. If they are sent, a broken connection or crash of one of the machines will be properly noticed. This also means that connections will die if the route is down temporarily.

**Rekey Interval**

Specify the number of **Seconds** or transferred **Bytes** after which the key exchange is done again.

If a value for both **Seconds** and **Bytes** is specified, rekeying is done whenever one of the values is reached, after which the counters are reset.

The defaults are `3600` seconds (1 hour) and `1000000000` bytes (~1 GB). The value `0` (zero) turns rekey requests off. This does not prevent the client from requesting rekeys.

**Encryption**

Under **Encryption**, select the **Ciphers** and **MACs** allowed for the connection from the list. To select several ciphers or MACs, hold down the **Shift** key while clicking.

## 4.1.9 Authentication

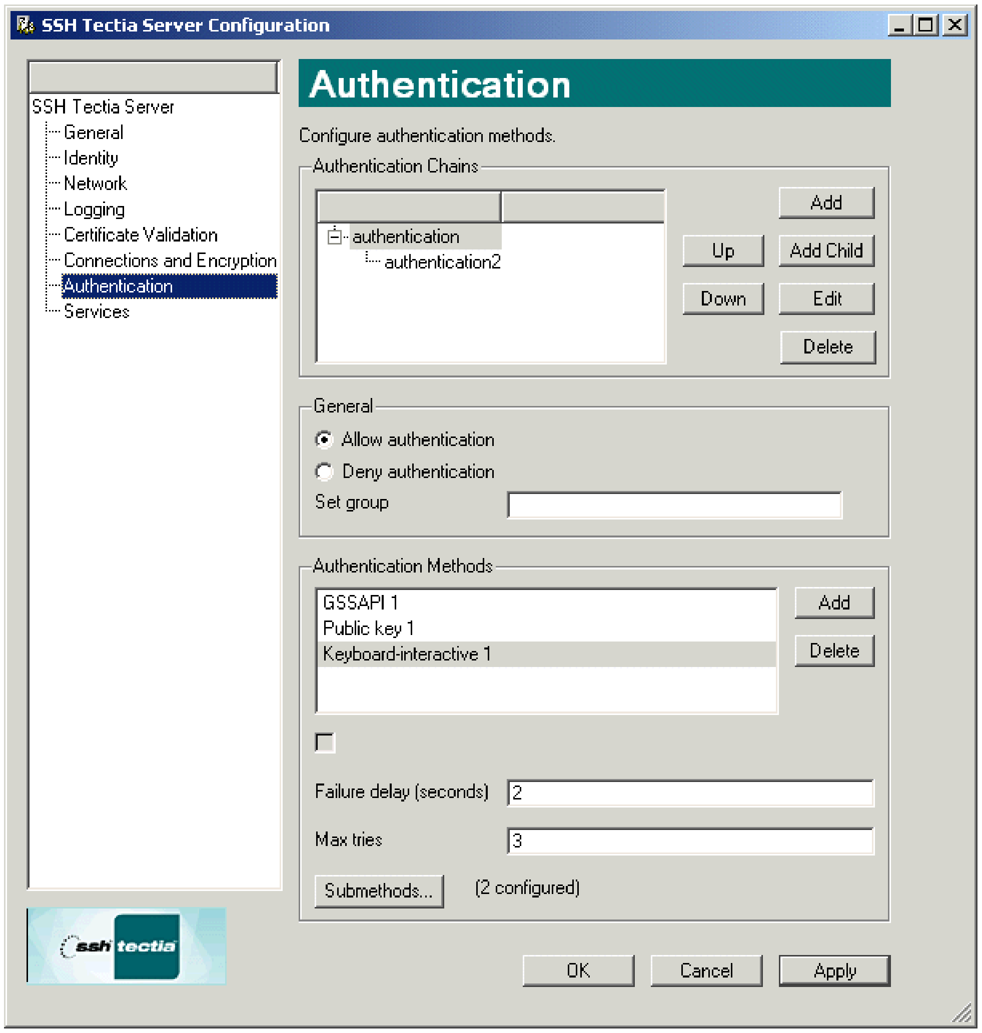
On the **Authentication** page you can configure the allowed and required user authentication methods.

Authentication options are specified as chains of authentication elements. An authentication element can include one or more selectors and different authentication methods. It may also include other authentication elements, forming an authentication chain.

Nesting authentication elements within each other sets the child methods as *required* (all must be passed for the authentication to be successful). Setting multiple `authentication` elements at the same level sets them as *optional* (one of the methods must be passed for the authentication to be successful).

The selectors define to which users an authentication element applies to. The order of the elements is important. For methods on the same level, the first matching method is used and the remaining methods are ignored. If the method has nested child methods, they are matched next using the same procedure.

If no selectors (or only empty selectors) are specified in an authentication element, the element matches to all users.



Authentication

Configure authentication methods.

Authentication Chains

authentication
authentication2

General

Allow authentication [✓]
Deny authentication

Set group

Authentication Methods

GSSAPI 1
Public key 1
Keyboard-interactive 1

Failure delay (seconds) 2

Max tries 3

(2 configured)

**Figure 4.10. SSH Tectia Server Configuration - Authentication page**

### Authentication Chains

To add a new authentication chain, click **Add** next to the **Authentication Chains** list. The **Selector** dialog box opens (Figure 4.7).

To change the order of the authentication elements, select an element from the list, and click **Up** and **Down** to move it. The methods are read in order and the first matching method on the list is used. If the method has nested child methods, they are matched next.

To add a nested authentication element, click **Add Child**. The **Edit Selector** dialog box opens. See Section 4.1.7 for information on editing the selectors.

To edit the selectors of an authentication element, select an element from the list and click **Edit**. The **Edit Selector** dialog box opens. See Section 4.1.7.

To delete an authentication element, select an element from the list and click **Delete**.

**General**

Select whether authentication is allowed or denied.

If an authentication chain ends in a deny action, the user is not allowed to log in. In a nested chain of authentication elements, it is possible, for example, to set the parent method to deny authentication and a child element with a selector to allow authentication. If the user matches the selector and successfully completes the authentication methods, login is allowed.

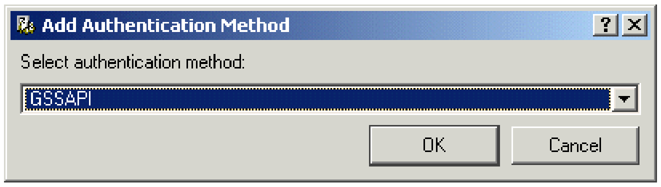
**Set group**

You can optionally enter a group name in the **Set group** field. This sets a group for the users that pass the particular authentication chain. The group definition can be later used when defining the allowed services for the user.

If the group is set here, it overrides any group definitions on the **Services** page. See Section 4.1.10.

**Authentication Methods**

To add an authentication method, click **Add** under **Authentication Methods**. A dialog box opens (Figure 4.11). Select the method from the list and click **OK**.



Select authentication method: GSSAPI

**Figure 4.11. Adding an authentication method**

The following authentication methods can be selected:

- **Password**
- **Public key**
- **Host-based**
- **Keyboard-interactive**
- **GSSAPI**

Depending on the authentication method, additional settings are available.

**Failure delay / Max tries**

For **Password** and **Keyboard-interactive** authentication, you can set the delay between failed attempts in seconds (**Failure delay**) and the maximum number of attempts (**Max tries**). The default delay is 2 seconds and default maximum is 3 attempts.

**Require DNS match**

For **Host-based** authentication, select the check box to require the hostname given by the client to match the one found in DNS. If the hostname does not match, the authentication fails.

**Allow ticket forwarding**

For **GSSAPI** authentication, select the check box to allow forwarding the Kerberos ticket over several connections.

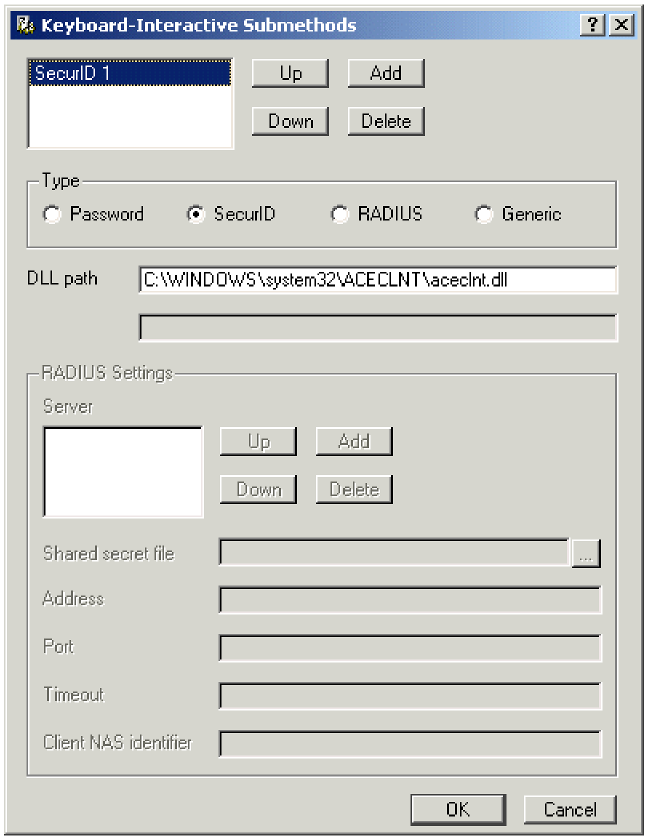
**DLL path**

For **GSSAPI** authentication, **DLL path** specifies the location of the necessary GSSAPI libraries. If this is not specified, the libraries are searched for in a number of common locations.

**Submethods**

For **Keyboard-interactive** authentication, several submethods can be specified.

To edit the submethods, click the **Submethods** button. A dialog box opens (Figure 4.12).



Type: Password [✓ SecurID] RADIUS Generic

DLL path: C:\WINDOWS\system32\ACECLNT\aceclnt.dll

**Figure 4.12. Keyboard-interactive submethods**

To add a submethod, click **Add** and select the **Type** of the submethod below.

The following submethods can be selected:

- **Password**
- **SecurID**
- **RADIUS**
- **Generic**

Depending on the submethod, additional settings are available.

To change the order of the submethods, select a submethod from the list, and click **Up** and **Down** to move it. The submethods are tried in the specified order.

To remove a submethod, select a method from the list and click **Delete**.

**DLL Path**
For the **SecurID** submethod, enter the path to the SecurID DLL.

**Name**
For the **Generic** submethod, enter the name of the method.

**Params**
For the **Generic** submethod, enter the initialization parameters.

**RADIUS Settings**
For the **RADIUS** submethod, click **Add** to add a RADIUS server.

For each RADIUS server, define a **Shared secret file**, server IP **Address**, **Port**, **Timeout**, and **Client NAS identifier**.

To change the order of the RADIUS servers, select a server from the list, and click **Up** and **Down** to move it. The servers are tried in the specified order.

To remove a RADIUS server, select a server from the list and click **Delete**.

### 4.1.10 Services

On the **Services** page you can set restrictions on the services (e.g. terminal, tunneling, SFTP) that the server provides.

Selectors can be used to define groups for users and different services can be allowed to each group.

The order of the groups is important. The user is put to the first matching group and the remaining groups are ignored.

If no selectors (or only empty selectors) are specified in a group element, the group matches to all users. However, creating a group like this is not sensible.

If the user was already put to a group during authentication (using **Set group**), the group definitions on the **Services** page are ignored.

The order of rules is also important. Rules for specified groups can be in any order (because they must match exactly to the group names), but the last rule should be a general rule, with no group definition. This rule is used for all users that do not have a group.

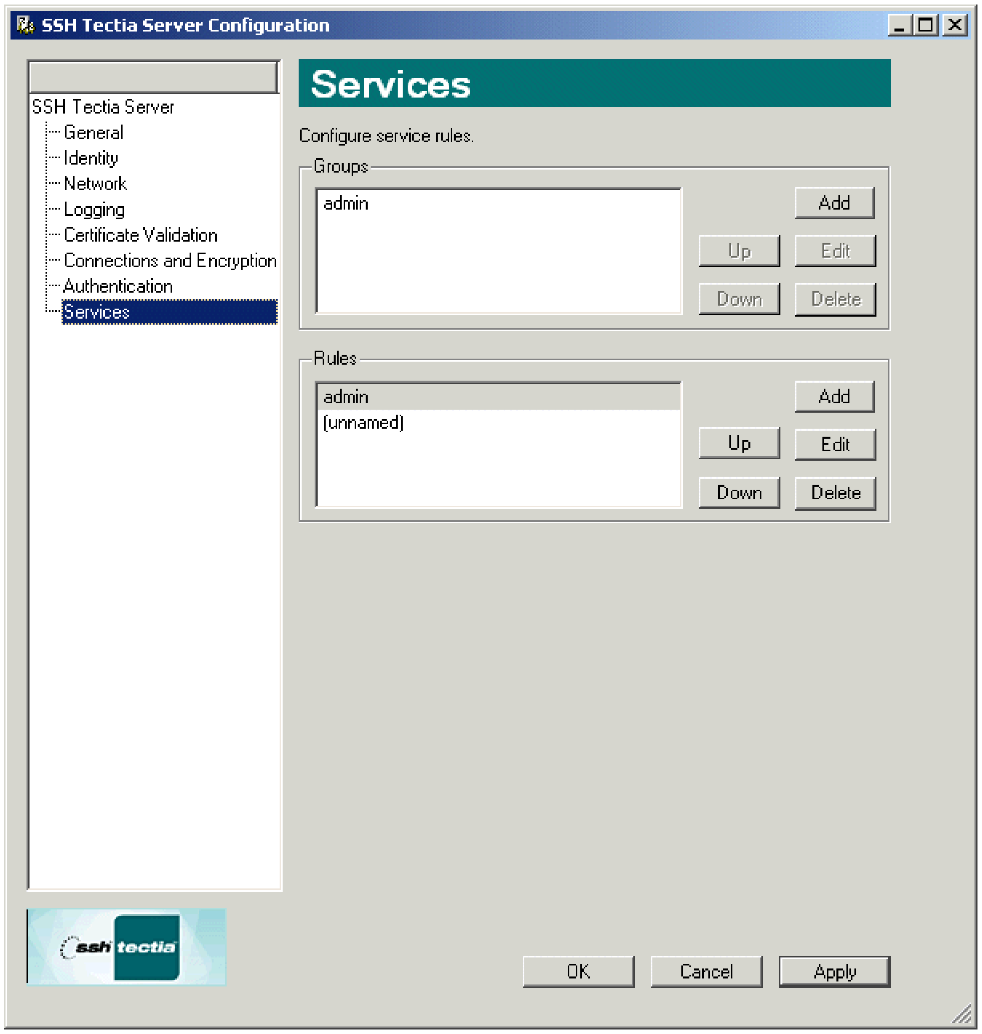


**Figure 4.13. SSH Tectia Server Configuration - Services page**

## Groups

To add a new group, click **Add** next to the **Groups** list. The **Selector** dialog box opens (Figure 4.7).

To change the order of the group elements, select an element from the list, and click **Up** and **Down** to move it. The groups are read in order and the user is put to the first matching group. If none of the groups match, the user will not have a group set.

To edit the selectors of a group element, select an element from the list and click **Edit**. The **Edit Selector** dialog box opens. See Section 4.1.7.

To delete a group element, select an element from the list and click **Delete**.

**Rules**

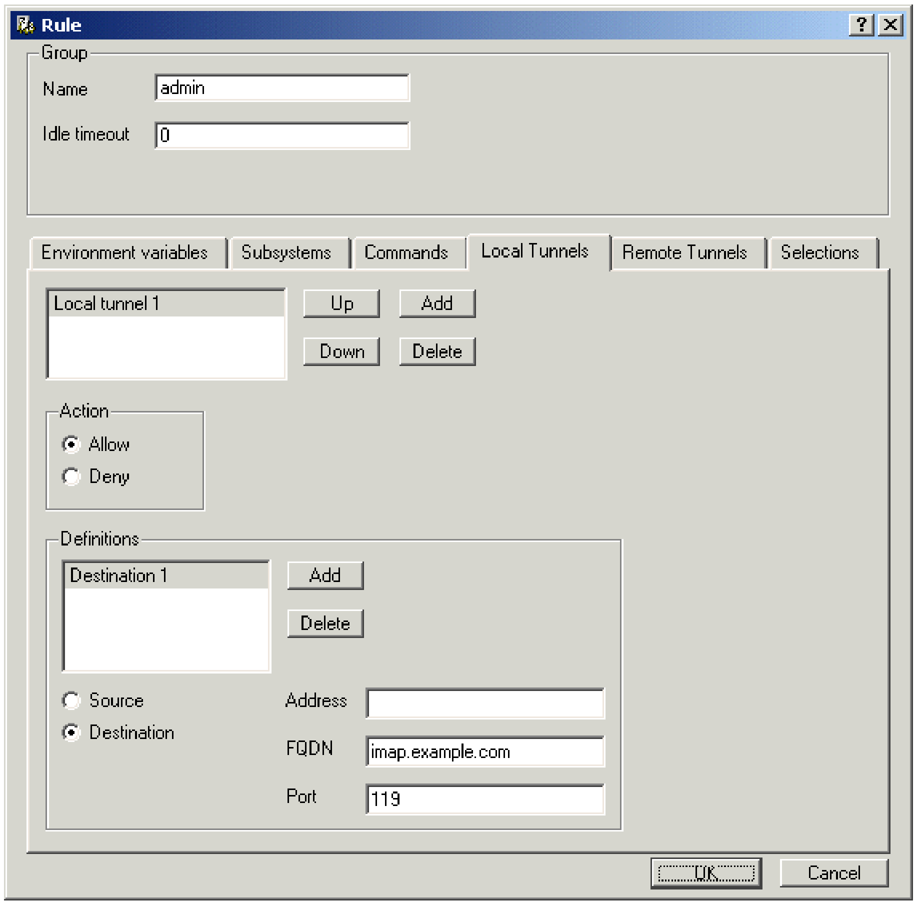
A default unnamed rule allows all users access to all services. The unnamed rule should be kept as the last rule, so it will apply to users that are not set in any group, but you should edit the rule according to your security policy.

To add a new rule, click **Add** next to the **Rules** list. The **Rule** dialog box opens (Figure 4.14).

To change the order of the rules, select a rule from the list, and click **Up** and **Down** to move it. The rules are read in order and the rule set that matches the user's group is used. The **(unnamed)** rule should be the last on the list.

To edit a rule, select a rule from the list and click **Edit**. The **Rule** dialog box opens (Figure 4.14).

To delete a rule element, select an element from the list and click **Delete**.



Name: admin

Idle timeout: 0

Local tunnel 1

Action: [✓ Allow] Deny

Destination 1

Source [✓ Destination]

Address:

FQDN: imap.example.com

Port: 119

**Figure 4.14. The Rule dialog box - the Local Tunnels tab**

**Group**

Enter the **Name** of the group to which this rule set applies.

The **Idle timeout** field sets the idle timeout limit in seconds. If the connection has been idle (all channels) this long, the connection is closed. Default is `0` (zero), which disables idle timeouts.

**Environment Variables**

On the **Environment Variables** tab, you can define the environment variables the user group can set. The variables are given in the **Allowed** field as a comma-separated list.

Allowed variables are normally matched case-insensitively. Alternatively, the variables can be specified in the **Allowed case-sensitive** field.

**Subsystems**

On the **Subsystems** tab, you can define the allowed and denied subsystems.

To add a subsystem, click **Add**. Enter the subsystem **Type** and select whether the subsystem is **Allowed**.

Define also the **Application**.

To remove a subsystem, click **Delete**.

The subsystem can contain several **Attributes**.

**Attributes**

To add an attribute, click **Add**.

Enter the **Name** and **Value** of the attribute.

To remove an attribute, click **Delete**.

#### Commands

On the **Commands** tab, you can define shell commands as allowed, denied, or forced.

To add a command rule, click **Add**.

To delete a command rule, select a rule form the list and click **Delete**.

- If the **Deny** action is set, all shell commands are denied and no further attributes should be specified.
- For the **Allow** and **Force** actions, the **Application** must be given. If the application is not given for the **Allow** action, all commands are allowed.
- If the **Force** action is set, the specified application is run automatically when the user logs in.

Applications are normally matched case-insensitively. Alternatively, the application can be specified using the **Application case-sensitive** field.

#### Local Tunnels

On the **Local Tunnels** tab, you can define rules for local TCP tunnels (port forwarding).

To add a tunnel rule, click **Add**.

To delete a tunnel rule, select a rule form the list and click **Delete**.

To change the order of the rules, select a rule from the list, and click **Up** and **Down** to move it. The rules are read in order and the first matching rule is used.

The tunneling **Action** can be **Allow** or **Deny**.

For more information on tunneling, see Section 8.2.

#### Definitions

Under **Definitions**, you can fine-tune the tunneling settings by adding source and destination restrictions.

To add a source address definition, click **Add** and select **Source**.

To add a destination address definition, click **Add** and select **Destination**.

The IP **Address** or the **FQDN** (not both) and the **Port** can be given for the **Source** and **Destination** definitions.

To delete a tunnel rule, select a rule form the list and click **Delete**.

### Remote Tunnels

On the **Remote Tunnels** tab, you can define rules for remote TCP tunnels (port forwarding).

To add a tunnel rule, click **Add**.

To delete a tunnel rule, select a rule form the list and click **Delete**.

To change the order of the rules, select a rule from the list, and click **Up** and **Down** to move it. The rules are read in order and the first matching rule is used.

The tunneling **Action** can be **Allow** or **Deny**.

For more information on tunneling, see Section 8.3.

#### Definitions

Under **Definitions**, you can fine-tune the tunneling settings by adding source and listener restrictions.

To add a source address definition, click **Add** and select **Source**.

To add a listener address definition, click **Add** and select **Listen**.

The IP **Address** or the **FQDN** (not both) and the **Port** can be given for the **Source** definition.

The IP **Address** and the **Port** can be given for the **Listen** definition.

To delete a tunnel rule, select a rule form the list and click **Delete**.

### Terminal

The **Allow terminal** setting defines whether terminal access is allowed or denied for the user group.

If terminal access is denied, also shell commands are denied, unless the commands are specifically allowed on the **Commands** tab.

Click **OK** to the accept the changes and return to the main **SSH Tectia Server Configuration** page.

# Chapter 5 Authentication

The Secure Shell protocol used by the SSH Tectia client/server solution provides mutual authentication – the client authenticates the server and the server authenticates the client. Both parties are assured of the identity of the other party.

The remote SSH Tectia Server host can authenticate itself using either traditional public-key authentication or certificate authentication.

Different methods can be used to authenticate SSH Tectia Client users. These authentication methods can be combined or used separately, depending on the level of functionality and security you want.

User authentication methods used by the client by default are, in the following order: GSSAPI, public-key, keyboard-interactive, and password authentication (if available). Public-key and certificate authentication are combined into the public-key authentication method.

The server allows GSSAPI, public-key, keyboard-interactive, and password authentication by default.

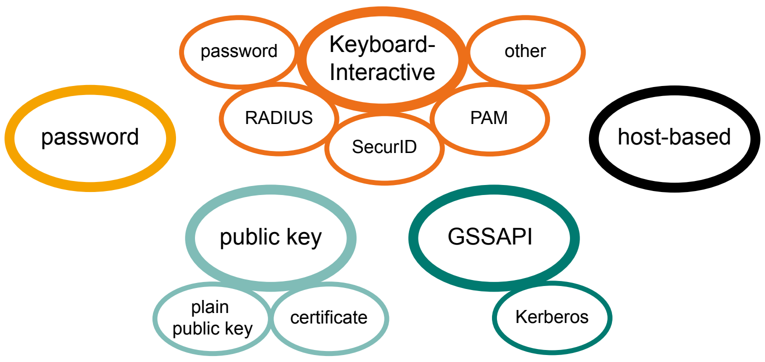


**Figure 5.1. User authentication methods**

## 5.1 Server Authentication with Public Keys

At the beginning of the connection, the server sends its public host key to the client for validation.

The key pair used for server authentication is defined on the server in the `ssh-server-config.xml` file with the following elements:

```
<params>
  <hostkey>
    <private file="/etc/ssh2/hostkey" />
    <public file="/etc/ssh2/hostkey.pub" />
  </hostkey>
...
</params>
```

Giving the public key in the configuration file is not mandatory. It will be derived from the private key if it is not found otherwise. Specifying the public key will, however, decrease start-up time for the software, as deriving the public key is a somewhat time-consuming operation.

During the installation process, one RSA key pair (with the file names `hostkey` and `hostkey.pub`) is generated and stored in the `/etc/ssh2` directory on Unix and in the `C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Server` directory on Windows. By default this key pair is used for server authentication.

Each SSH Tectia Server can have multiple host keys. You could have, for example, the following set of parameters in your `ssh-server-config.xml` file:

```
<params>
  <hostkey>
    <private file="/etc/ssh2/hostkey_dsa" />
    <public file="/etc/ssh2/hostkey_dsa.pub" />
  </hostkey>

  <hostkey>
    <private file="/etc/ssh2/hostkey_rsa" />
    <public file="/etc/ssh2/hostkey_rsa.pub" />
  </hostkey>
...
</params>
```

Both keys are stored in memory when the `ssh-server-g3` process is started, which means that either one of them can be used to authenticate the server.

We recommend that you use a maximum of one DSA and one RSA key pair. If also certificates are used in server authentication, an additional two host key pairs (DSA with certificate and RSA with certificate) can be used for a total of four host keys.

On Windows, the host keys can be configured with the **SSH Tectia Server Configuration** tool on the **Identity** page. See Section 4.1.3.

### 5.1.1 Generating the Host Key

The host public-key pair (1536-bit RSA) is generated during the installation of SSH Tectia Server. You only need to regenerate it if you want to change your host key pair.

The command-line tool `ssh-keygen-g3` can be used to generate the host key pair. It can be used for creating the user key pairs as well.

On Unix, to (re)generate the host key, give the following command with root privileges:

```
# ssh-keygen-g3 -P -H hostkey
```

On Windows, to (re)generate the host key, give the following command:

```
ssh-keygen-g3.exe -P -H hostkey
```

This will generate a 2048-bit DSA key pair (without a passphrase) and save it in the default host key directory (`/etc/ssh2` on Unix, `C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Server` on Windows) with the names `hostkey` and `hostkey.pub`. For more information on the key generation options, see ssh-keygen-g3(1).

After the new key pair has been created, run `ssh-server-config-tool` to reconfigure the server. See ssh-server-config-tool(8).

#### Note

It is of utmost importance that the private key is *never* readable by anyone but the server itself. Store the private key of the server in a safe directory where access is denied for all users.

### 5.1.2 Notifying the Users of the Host Key Change

Administrators that have other users connecting to their server should notify the users of the host key change. If you do not, the users will receive a warning the next time they connect because the host key the users have saved on their disk for your server does not match the host key now being actually provided by your server. The users may not know how to respond to this error. SSH Tectia Manager (available separately) provides an automatic mechanism for distributing the host keys.

You can run `ssh-keygen-g3` to generate a fingerprint for your new public host key which you can provide to your users via some unalterable method (for example, by a digitally signed e-mail or by displaying the fingerprint on secured bulletin board).

On Unix, the command for generating the fingerprint is:

```
# ssh-keygen-g3 -F hostkey.pub
```

On Windows, the command is:

```
ssh-keygen-g3.exe -F hostkey.pub
```

When the users connect and receive the error message about the host key having changed, they can compare the fingerprint of the new key with the fingerprint you have provided in your e-mail, and ensure that they are connecting to the correct SSH Tectia Server. Inform your users to notify you if the fingerprints do not match, or if they receive a message about a host key change and do not receive a corresponding message from you notifying them of the change.

This procedure can help ensure that you do not become a victim of a man-in-the-middle attack, as your users will notify you if the host key fingerprints do not match. You will also be aware if the users encounter host key change messages when you have not regenerated your host key pair.

It is also possible to send the public host key to the users via an unalterable method. The users can save the key in the `$HOME/.ssh2/hostkeys` directory on Unix or in the `%USERPROFILE%\Application Data\SSH\HostKeys` directory on Windows as `key_<port>_<host>.pub` (for example, `key_22_banana.ssh.com.pub`). In this case, a manual fingerprint check is not needed. SSH Tectia Manager can distribute the host keys to these directories automatically.

## 5.2 Server Authentication with Certificates

Server authentication with certificates happens similarly to server authentication with public keys, except that the possibility of a man-in-the-middle attack during the first connection to a particular server is eliminated. The signature of a certification authority in the server certificate guarantees the authenticity of the server certificate even in the first connection.

A short outline of the server authentication process with certificates is detailed below:

1. The server sends its certificate (which contains a public key) to the client. The packet also contains random data unique to the session, signed by the server's private key.

2. As the server certificate is signed with the private key of a certification authority (CA), the client can verify the validity of the server certificate by using the CA certificate.

3. The client checks that the certificate matches the name of the server. (This check can be disabled by setting the `end-point-identity-check` attribute of the `cert-validation` element in the client configuration file to `no`.)

4. The client verifies that the server has a valid private key by checking the signature in the initial packet.

During authentication the system checks that the certificate has not been revoked. This can be done either by using the Online Certificate Status Protocol (OCSP) or a Certificate Revocation List (CRL), which can be published either in an LDAP or HTTP repository.

OCSP is automatically used if the certificate contains a valid **Authority Info Access** extension, or an OCSP responder has been separately configured. If no OCSP responder is available or the OCSP validation produces a negative result, the CRLs are automatically used. If LDAP is used as the CRL publishing method, the LDAP repository location can also be defined in the `ssh-broker-config.xml` file. See *SSH Tectia Client User Manual* for more information.

### 5.2.1 Certificate Enrollment Using `ssh-cmpclient`

Certificates can be enrolled using the `ssh-cmpclient` command-line tool (`ssh-cmpclient.exe` on Windows).

To configure SSH Tectia Server to authenticate itself using X.509 certificates, perform the following tasks:

1. Enroll a certificate for the server.

   This can be done with the `ssh-cmpclient` command-line tool, for example:

   ```
   $ ssh-cmpclient INITIALIZE \
     -P generate://ssh2@rsa:1024/hostcert_rsa \
     -o /etc/ssh2/hostcert_rsa \
     -p 62154:ssh \
     -s "C=FI,O=SSH,CN=testserv;dns=testserv.ssh.com" \
     http://pki.ssh.com:8080/pkix/ \
     'C=FI, O=SSH Communications Security Corp, CN=Secure Shell Test CA'
   ```

   Note that the DNS address parameter (`dns`) needs to correspond to the fully qualified domain name of the server.

   Remember to define also the SOCKS server (`-S`) before the CA URL, if required.

   For more information on the `ssh-cmpclient` syntax, see ssh-cmpclient-g3(1).

2. Define the private key and the server certificate in the `ssh-server-config.xml` file:

   ```
   <params>
     <hostkey>
       <private file="/etc/ssh2/hostcert_rsa" />
       <x509-certificate file="/etc/ssh2/hostcert_rsa.crt" />
     </hostkey>
   ...
   </params>
   ```

   Alternatively, on Windows when using the **SSH Tectia Server Configuration** tool, enter the private key and certificate filenames on the **Identity** page. See Section 4.1.3.

3. Run `ssh-server-config-tool` to take the new configuration in use. See ssh-server-config-tool(8).

   On Windows, click **Save** to take the new settings in use.

## 5.3 Server Authentication using External Host Keys

In addition to conventional keys and certificates stored as files on disk, several external key providers are available for accessing keys and certificates stored in hardware tokens or external software modules. The example below initializes the `software` external key provider, which is used to access keys and certificates on disk, and instructs it to read all keys in `/etc/ssh2/hostkeys`.

```
<params>
  <hostkey>
    <externalkey type="software"
      init-info="directory(/etc/ssh2/hostkeys)"/>
  </hostkey>
...
</params>
```

Each `hostkey` element can be used for setting up one external key provider. Each provider may provide any number of keys to the server. It should be noted that due to the limitations of the SSH2 protocol, having more than one key of each type (RSA, DSA, X.509 certificate with RSA key and X.509 certificate with DSA key) is discouraged.

## 5.4 User Authentication with Passwords

The password authentication method is the easiest to implement, as it is set up by default. Since all communication is encrypted, passwords are not available for eavesdroppers.

On a Unix system, password authentication uses the `/etc/passwd` or `/etc/shadow` file, depending on how the passwords are set up.

On Windows, password authentication uses the Windows password to authenticate the user at login time.

To enable password authentication on the server, the `authentication-methods` element of the `ssh-server-config.xml` file must contain an `auth-password` element. For example:

```
<authentication-methods>
  <authentication action="allow">
    ...
    <auth-password failure-delay="2" max-tries="3" />
  </authentication>
</authentication-methods>
```

Also other authentication methods can be allowed. Place the least interactive method first (password is usually the last one).

By using selectors, it is possible to allow or require password authentication only for a specified group of users. See the section called “Selectors” for more information.

On Windows, using the **SSH Tectia Server Configuration** tool, password authentication can be allowed on the **Authentication** page. See Section 4.1.9.

**Note**

With passwords, it is also possible to use keyboard-interactive authentication. See Section 5.8.1 for more information.

### 5.4.1 Special Considerations on Windows

Please note that if domain user accounts are used, entering the password is always required. Non-interactive login is possible only for local accounts when public-key authentication is used.

SSH Tectia Server does not need a user management program of its own – the user accounts are created with the standard Windows User Manager.

SSH Tectia Server can use either local or domain user accounts. However, for domain user accounts, the password authentication method is always required. If a user attempts a public-key login with a domain account, an error is displayed. Domain accounts should be specified using the `domain\user` notation.

**Note**

User login requires the *log on locally* right. On domain controllers, this right is disabled by default. If SSH Tectia Server has been installed on a domain controller, the *log on locally* permission must be enabled on the domain controller for the *Domain Users* group.

## 5.5 User Authentication with Public Keys

Public-key authentication is based on the use of digital signatures and provides the best authentication security. To use public-key authentication, the user must first create a key pair on the client, and upload the public key to the server. The default directory for the user's public keys is `$HOME/.ssh2/authorized_keys` on Unix and `%USERPROFILE%\.ssh2\authorized_keys` on Windows.

To enable public-key authentication on the server, the `authentication-methods` element of the `ssh-server-config.xml` file must contain an `auth-publickey` element. For example:

```
<authentication-methods>
  <authentication action="allow">
    <auth-publickey />
    ...
  </authentication>
</authentication-methods>
```

Also other authentication methods can be allowed. Place the least interactive method first.

By using selectors, it is possible to allow or require public-key authentication only for a specified group of users. See the section called “Selectors” for more information.

On Windows, using the **SSH Tectia Server Configuration** tool, public-key authentication can be allowed on the **Authentication** page. See Section 4.1.9.

### 5.5.1 Special Considerations on Windows

On Windows, please note that public-key authentication works only with local user accounts. Local accounts are unable to access the network resources (such as computers or printers) of any domain.

To use public-key authentication, the public key must be first created on the client, and uploaded to the server.

On the server, the user's public keys must be located under the user's profile directory (use the `%D` pattern string when specifying the user key directory). The recommended location is the `%USERPROFILE%\.ssh2\authorized_keys` directory. This location reflects the standard Unix usage and works with the default settings of SSH Tectia Client and the user's profile directory has the appropriate access permissions (set by the operating system during the account creation).

**Note**

If the root directory for SFTP has been changed (chrooted) to some other location than the user profile directory, using public-key authentication will require changing the user key directory setting accordingly. Uploading the public key will not succeed if SFTP is set to use some other directory than the specified location for public keys.

## 5.6 User Authentication with Certificates

Certificate authentication is technically a part of the the public-key authentication method. The signature created with the private key and the verification of the signature using the public key (contained in the X.509 certificate when doing certificate authentication) are done identically with conventional public keys and certificates. The major difference is in determining whether a specific user is allowed to log in with a specific public key or certificate. With conventional public keys, every server must have every user's public key, whereas with certificates the users' public keys do not have to be distributed to the servers - distributing the public key of the CA (self-signed certificate) is enough.

In brief, certificate authentication works in the following way:

1. The client sends the user certificate (which includes the user's public key) to the server. The packet also contains random data unique to the session and signed by the user's private key.

2. The server uses the CA certificate (and external resources as required) to check that the user's certificate is valid.

3. The server verifies that the user has a valid private key by checking the signature in the initial packet.
4. The server matches the user certificate against the rules in the server configuration file to decide whether login is allowed or not.

Compared to traditional public-key authentication, this method is more secure because the system checks that the user certificate was issued by a trusted CA. In addition, certificate authentication is more convenient because no local database of user public keys is required on the server.

It is also easy to deny a user's access to the system by revoking his or her certificate, although this doesn't take effect until the next CRL update and requires that every other authentication method has been disabled. The status of a certificate can be checked either by using the Online Certificate Status Protocol (OCSP) or certificate revocation lists (CRLs), which can be published either in an LDAP or HTTP repository.

OCSP is used if the certificate contains a valid **Authority Info Access** extension or if an `ocsp-responder` has been defined in the `ssh-server-config.xml` file. If OCSP fails or is not defined, CRLs are used. The certificate should contain a valid **CRL Distribution Point** extension or an LDAP server for CRL fetching should be defined in the `ssh-server-config.xml` file.

## 5.6.1 Certificate Configuration

To configure the server to allow user authentication with X.509 certificates, perform the following tasks:

1. Acquire the CA certificate and copy it to the server machine. You can either copy the X.509 certificate(s) as such or you can copy a PKCS #7 package including the CA certificate(s).

   Certificates can be extracted from a PKCS #7 package by specifying the `-7` flag with `ssh-keygen-g3`.

2. Specify the CA certificate and the CRL and OCSP settings in the `ssh-server-config.xml` file. An example is shown below:

```
<params>
  ...
  <cert-validation socks-server-url="socks://fw.example.com:1080">
    <ldap-server address="ldap.example.com" port="389" />
    <ocsp-responder validity-period="60" url="https://ca.example.com/ocsp-1" />
    <cert-cache-file file="/var/cert-cache.dat" />
    <crl-auto-update update-before="30" minimum-interval="600" />
    <crl-prefetch interval="1800" url="http://ca.example.com/default.crl" />
    <dod-pki enable="no" />
    <ca-certificate name="exa-ca1" file="/etc/ssh2/exa-ca1.crt" />
  </cert-validation>
</params>
```

   You can define several CA certificates by using several `ca-certificate` elements. The server will accept only certificates issued by defined CA(s). Only the ca-certificates are mandatory, all other configuration items featured above are just examples that may be used as needed.

The SOCKS server must be defined if the OCSP and CRL (LDAP) services are located behind a firewall.

On Windows, using the **SSH Tectia Server Configuration** tool, the corresponding settings can be made on the **Certificate Validation** page. See Section 4.1.6.

3. Certificate authentication is a part of the `publickey` authentication method. Enable public-key authentication in the `ssh-server-config.xml` file and create rules that specify which certificates authorize logging into which accounts.

   The following is an example of certificate authentication rules in the `ssh-server-config.xml` file:

```
<authentication-methods>
  <authentication action="allow">
    <auth-publickey />
    <authentication action="allow">
      <selector>
        <certificate field="ca-list" pattern="exa-ca1,exa-ca2" />
        <certificate field="issuer-name" pattern="C=FI, O=SSH, CN=*" />
        <certificate field="subject-name" pattern="C=FI, O=SSH, CN=%username%" />
        <certificate field="serial-number" pattern="123456" />
        <certificate field="altname-email" pattern="%username%@ssh.com" />
        <certificate field="altname-fqdn" pattern="client.ssh.com" />
        <certificate field="altname-upn" pattern="%username%@ssh" />
        <certificate field="altname-ip" pattern="10.2.3.5" />
      </selector>
    </authentication>
    <authentication action="deny" />
  </authentication>
  ...
</authentication-methods>
```

   In this example, as the last action, access is denied for all users whose certificates were not explicitly allowed. This is not strictly needed, since the server automatically inserts a authentication block named `implicit-certificate-deny` after other blocks to catch all certificate authentications that do not match anything else.

   Certificate authentication can be restricted using the following `field` attributes:

   - `ca-list`: The pattern is a comma-separated list of CA names. The names that are defined in the `ca-certificate` element are used.
   - `issuer-name`: The pattern is the required certificate issuer name in LDAP DN (distinguished name) string format. The issuer name may contain glob patterns ('*' and '?') but only in the component values, not names. For example, `"C=FI, O=SSH, CN=*"` is a legal pattern, but `"C=FI, *=SSH, CN=TestCA"` is not).
   - `subject-name`: The pattern is the required subject name in LDAP DN (distinguished name) string format. Matching is done in similar manner as with the issuer name described above.

- `serial-number`: The pattern is the required serial number of the certificate. A combination of issuer name and serial number can be used to uniquely identify a certificate.
- `altname-email`: The pattern is the e-mail address that must be present in the certificate as a subject alternative name.
- `altname-upn`: The pattern is the principal name that must be present in the certificate as a subject alternative name.
- `altname-ip`: The pattern is the IP address that must be present in the certificate as a subject alternative name. Also a range of addresses can be given (for example, `10.1.0.11-10.1.0.61` or `10.1.0.0/8`).
- `altname-fqdn`: The pattern is a list of fully qualified domain names (FQDN) that may contain glob patterns ('*' and '?'), one of which must match with a subject alternative name of type FQDN in the certificate.

The patterns of type `subject-name`, `issuer-name`, `altname-email` and `altname-upn` can also contain special strings which are processed before comparing the pattern with the user's certificate. These strings are `%username%` (user's login name), `%homedir%` (user's home directory), and `%hostname%` (the name of the host the user is logging from, reverse mapped from the IP).

On Windows, using the **SSH Tectia Server Configuration** tool, certificate authentication rules can be configured on the **Authentication** page. See Section 4.1.9.

4. Run `ssh-server-config-tool` to take the new configuration in use. See ssh-server-config-tool(8).

   On Windows, click **Save** to take the new settings in use.

## 5.7 Host-Based User Authentication

Host-based authentication uses a public host key of the client machine to authenticate a user to the remote server. Host-based authentication can be used with SSH Tectia Client on Unix. The SSH Tectia Server can be either an Unix or Windows server. Usually also SSH Tectia Server is installed on the client machine. This provides a non-interactive form of authentication, and is best used in scripts and automated processes, such as cron jobs. Host-based authentication can be used to automate backups and file transfers, or in other situations where a user will not be present to input authentication information.

The nature of any non-interactive login is inherently insecure. Whenever authentication without user challenge is permitted, some level of risk must be assumed. If feasible, public-key authentication is preferred. SSH Tectia Server provides host-based authentication as a form of non-interactive login that is more secure than the `.rhosts` method used by the Berkeley 'r' commands, but it cannot resolve the inherent lack of security of non-interactive logins.

This means that you should take aggressive measures to ensure that any client machine set up for host-based authentication is adequately secured, both by software and hardware, to prevent unauthorized logins to your server.

Host-based authentication can be enabled either by using traditional public keys or by using certificates.

In the following instructions, `Server` is the SSH Tectia Server to which you are trying to connect. `ServerUser` is the username on the server that you are logging into. `Client` is the machine running an SSH Tectia Client. `ClientUser` is the username on the client machine that should be allowed to log in to `Server` as `ServerUser`.

## 5.7.1 Using Traditional Public Keys

### Client Configuration

To enable host-based authentication with traditional public keys on the client, do the following as `ClientUser`:

1. Generate a host key. If SSH Tectia Server has been installed on the same machine, the host key pair `/etc/ssh2/hostkey` and `/etc/ssh2/hostkey.pub` has been generated during installation and you can skip this step. Otherwise, give the following command:

   ```
   # ssh-keygen-g3 -P /etc/ssh2/hostkey
   ```

   Optionally, you can define a custom location or name for the host key in the `ssh-server-config.xml` file. If SSH Tectia Server is not installed on the client host, you can create the configuration file manually and save it in the `/etc/ssh2` directory.

2. Add the following line in the `ssh-broker-config.xml` file:

   ```
   <authentication-methods>
     <authentication-method name="hostbased" />
     ...
   </authentication-methods>
   ```

   Also other authentication methods can be listed.

### Server Configuration

Do the following as the server administrator:

1. Copy the client's `/etc/ssh2/hostkey.pub` file over to the server. Note that this requires root permissions on the client, and optionally on the server as well.

   SSH Tectia Server looks for the host keys to use for host-based authentication in the `/etc/ssh2/trusted_hosts` directory on Unix and in the `C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Server\trusted_hosts` directory on Windows.

   You have to name the client's public key as follows on the server:

```
client.example.com.ssh-dss.pub
```

In the example, `client.example.com` is the hostname the client is sending to the server. When the server receives the client's public key, it forms a path based on the hostname and the key type (`ssh-dss` or `ssh-rsa`) and compares the received public key to the key on the disk. If the public key matches and the user's login name in the remote end matches the name the user is trying to log in on the server, the user is let in after the signature check.

2. To enable host-based authentication on the server, in the `ssh-server-config.xml` file, under the `authentication-methods` element, add an `auth-hostbased` element:

   ```
   <authentication-methods>
     <authentication action="allow">
       <auth-hostbased require-dns-match="no" />
       ...
     </authentication>
   </authentication-methods>
   ```

   If you want to allow multiple authentication methods, place the least interactive method first (this means usually the host-based method).

   To force an exact match between the hostname that the client sends to the server and the client's reverse mapped DNS entry, set the `require-dns-match` attribute to `yes`.

   In this case, make sure the `/etc/hosts` file has the fully qualified domain name listed before the short hostname, for example:

   ```
   123.123.123.123   client.example.com   client
   ```

   Even if you are not using `/etc/hosts` as your primary resolver, you may need to add entries to it for the client and the server to allow them to resolve each other's fully qualified domain names (if they are not able to do so otherwise).

   Please note that when exact DNS matching is set as required, host-based authentication through NAT (Network Address Translation) will not work.

   On Windows, using the **SSH Tectia Server Configuration** tool, host-based authentication can be configured on the **Authentication** page. See Section 4.1.9.

3. Run `ssh-server-config-tool` to take the new configuration in use. See ssh-server-config-tool(8).

   On Windows, click **Save** to take the new settings in use.

To test that host-based authentication works, log in to `Client` as `ClientUser` and run the following command:

```
$ sshg3 ServerUser@server uptime
```

You should get back the results of uptime on the server.

## 5.7.2 Using Certificates

It is possible to use a certificate instead of the traditional public-key pair to authenticate the client host.

The endpoint identity check, where the server verifies that the certificate actually belongs to the client that is attempting host-based authentication, is performed according to the following rules:

1. One of the DNS subject alternative names in the client certificate must match the client's fully qualified domain name obtained by doing a reverse lookup on the client's IP address. The alternative names may have an asterisk (`*`) as the first component, in which case only the domain part is checked.

2. If the client's IP address cannot be reverse-mapped, the IP address is compared to the certificate's IP subject alternative names.

3. If the above checks do not produce a positive result, the certificate's subject name is checked. If it has a CN component that matches the client's reverse-mapped fully qualified domain name or IP address, the certificate is accepted.

### Client Configuration

To enable host-based authentication with certificates on `Client`, do the following as `ClientUser`:

1. Add the following line in the `ssh-broker-config.xml` file:

```
<authentication-methods>
  <authentication-method name="hostbased" />
  ...
</authentication-methods>
```

2. Enroll a certificate for `Client`. See Section 5.6 for more information.

   The certificate must contain a `dns` extension which contains the fully qualified domain name (FQDN) of `Client`.

   Note that the private key associated with the certificate needs to be stored with an empty passphrase.

3. Define the private key and certificate in `ssh-server-config.xml` on `Client`:

```
<params>
  <hostkey>
    <private file="/etc/ssh2/hostcert" />
    <x509-certificate file="/etc/ssh2/hostcert.crt" />
  </hostkey>
...
</params>
```

   If SSH Tectia Server is not installed on `Client`, create the configuration file manually and save it in the `/etc/ssh2` directory.

### Server Configuration

Do the following as the server administrator:

1. Specify the CA certificate in the `ssh-server-config.xml` file:

   ```
   <cert-validation>
     <ca-certificate name="myca" file="/etc/ssh2/ca-certificate.crt" />
     ...
   </cert-validation>
   ```

2. In the `ssh-server-config.xml` file, under the `authentication-methods` element, add an `auth-hostbased` element:

   ```
   <authentication-methods>
     <authentication name="hostbased-block">
       <auth-hostbased require-dns-match="no" />
       <authentication action="allow" name="hostbased-cert-allow">
         <selector>
            <certificate field="ca-list" pattern="exa-ca1" />
         </selector>
       </authentication>
     <authentication action="deny" />
     </authentication>
   </authentication-methods>
   ```

   In addition to being signed by the required CA, the certificate must pass the endpoint identity check, described in detail in Section 5.7.2.

   If you want to allow multiple authentication methods, place the least interactive method first (this means usually the host-based method).

   On Windows, using the **SSH Tectia Server Configuration** tool, host-based authentication can be configured on the **Authentication** page. See Section 4.1.9.

3. Run `ssh-server-config-tool` to take the new configuration in use. See ssh-server-config-tool(8).

   On Windows, click **Save** to take the new settings in use.

## 5.8 User Authentication with Keyboard-Interactive

Keyboard-interactive is a generic authentication method that can be used to implement different types of authentication mechanisms. Any currently supported authentication method that requires only the user's input can be performed with keyboard-interactive.

Currently, the following keyboard-interactive submethods are supported:

- password

- PAM (Unix only, see note below)
- RSA SecurID
- RADIUS

Methods that require passing some binary information, such as public-key authentication, cannot be used as submethods of keyboard-interactive. But public-key authentication, for example, can be used as an additional method alongside keyboard-interactive authentication.

**Note**

PAM has support for binary messages and client-side agents, and those cannot be supported with keyboard-interactive.

The client cannot request any specific keyboard-interactive submethod if the server allows several optional submethods. The order in which the submethods are offered depends on the server configuration. However, if the server allows, for example, the two optional submethods SecurID and password, the user can skip SecurID by pressing enter when SecurID is offered by the server. The user will then be prompted for a password.

### 5.8.1 Password Submethod

Password authentication can also be used over keyboard-interactive.

The following example shows settings for keyboard-interactive authentication using the password submethod in the `ssh-server-config.xml` file:

```
<authentication-methods>
  <authentication action="allow">
  ...
    <auth-keyboard-interactive max-tries="3" failure-delay="2">
      <submethod-password />
    </auth-keyboard-interactive>
  ...
  </authentication>
</authentication-methods>
```

On Windows, using the **SSH Tectia Server Configuration** tool, keyboard-interactive authentication can be configured on the **Authentication** page. See Section 4.1.9.

### 5.8.2 Pluggable Authentication Module (PAM) Submethod

Pluggable Authentication Module is an authentication framework used in Unix systems. In SSH Tectia, support for PAM is enabled as a submethod of keyboard-interactive authentication.

When PAM is used, SSH Tectia Server transfers the control of authentication to the PAM library, which will then load the modules specified in the PAM configuration file. Finally, the PAM library tells SSH Tectia

Server whether or not the authentication was successful. SSH Tectia Server is not aware of the details of the actual authentication method employed by PAM. Only the final result is of interest.

The following example shows settings for keyboard-interactive authentication using the PAM submethod in the `ssh-server-config.xml` file:

```
<authentication-methods>
  <authentication action="allow">
  ...
    <auth-keyboard-interactive max-tries="3" failure-delay="2">
      <submethod-pam dll-path="path-to-pam-dll" />
    </auth-keyboard-interactive>
  ...
  </authentication>
</authentication-methods>
```

On Windows, using the **SSH Tectia Server Configuration** tool, keyboard-interactive authentication can be configured on the **Authentication** page. See Section 4.1.9.

**Note**

SSH Communications Security does not provide technical support on how to configure PAM. Our support only covers SSH Tectia applications.

### 5.8.3 RSA SecurID Submethod

RSA SecurID is a widely-used two-factor authentication method based on the use of SecurID Authenticator tokens. In SSH Tectia, support for RSA SecurID is enabled as a submethod of keyboard-interactive authentication.

To use SecurID authentication, you should be familiar with the operation of *RSA ACE/Server* (*RSA Authentication Manager*).

The following example shows settings for keyboard-interactive authentication using the SecurID submethod in the `ssh-server-config.xml` file:

```
<authentication-methods>
  <authentication action="allow">
  ...
    <auth-keyboard-interactive max-tries="3" failure-delay="2">
      <submethod-securid dll-path="path-to-securid-dll" />
    </auth-keyboard-interactive>
  ...
  </authentication>
</authentication-methods>
```

On Windows, using the **SSH Tectia Server Configuration** tool, keyboard-interactive authentication can be configured on the **Authentication** page. See Section 4.1.9.

**Note**

SSH Communications Security does not provide technical support on how to configure *RSA ACE/Server*. Our support only covers SSH Tectia applications.

### 5.8.4 RADIUS Submethod

RADIUS (Remote Authentication Dial-In User Service) is a protocol for checking a user's authentication and authorization information from a remote server. It was originally intended for authenticating dial-in users, but is also suitable for use with Secure Shell. In SSH Tectia, RADIUS is implemented as a submethod of keyboard-interactive authentication.

The following example shows settings for keyboard-interactive authentication using the RADIUS submethod in the `ssh-server-config.xml` file:

```
<authentication-methods>
  <authentication action="allow">
  ...
    <auth-keyboard-interactive max-tries="3" failure-delay="2">
      <submethod-radius>
        <radius-server address="10.1.61.128"
         port="1812" client-nas-identifier="nasid">
          <radius-shared-secret file="&configdir;/radius-secret-file" />
        </radius-server>
      </submethod-radius>
   </auth-keyboard-interactive>
  ...
  </authentication>
</authentication-methods>
```

On Windows, using the **SSH Tectia Server Configuration** tool, keyboard-interactive authentication can be configured on the **Authentication** page. See Section 4.1.9.

**Note**

SSH Communications Security does not provide technical support on how to configure RADIUS. Our support only covers SSH Tectia applications.

## 5.9 User Authentication with GSSAPI

GSSAPI (Generic Security Service Application Programming Interface) is a function interface that provides security services for applications in a mechanism-independent way. This allows different security mechanisms to be used via one standardized API. GSSAPI is often linked with Kerberos, which is the most common mechanism of GSSAPI.

For Windows, GSSAPI offers integrated authentication for both old NT4 networks with NTLM authentication and for new Windows 2000/2003 networks with Kerberos. This method utilizes domain accounts, since local accounts are not transferable across machine boundaries.

To enable GSSAPI authentication on the server, the `authentication-methods` element of the `ssh-server-config.xml` file must contain an `auth-gssapi` element. For example:

```
<authentication-methods>
  <authentication action="allow">
  ...
    <auth-gssapi dll-path="path-to-gssapi-dll" />
  ...
  </authentication>
</authentication-methods>
```

Also other authentication methods can be listed. Place the least interactive method first.

On Windows, using the **SSH Tectia Server Configuration** tool, GSSAPI authentication can be configured on the **Authentication** page. See Section 4.1.9.

### Note

SSH Communications Security does not provide technical support on how to configure Kerberos. Our support only covers SSH Tectia applications.

# Chapter 6 System Administration

Secure system administration is the most common use case for Secure Shell. This chapter describes typical system administration settings and available auditing options of SSH Tectia Server and gives an example of an SSH Tectia Server (A) configuration.

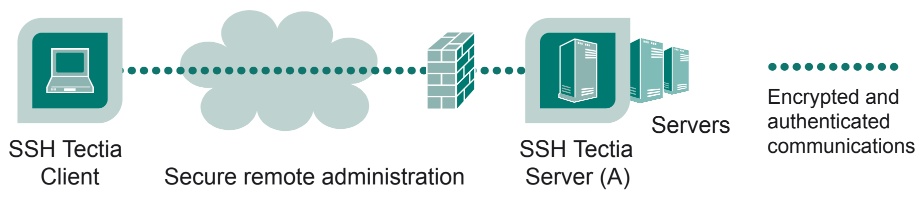

SSH Tectia Client
Secure remote administration
SSH Tectia Server (A)
Servers
Encrypted and authenticated communications

**Figure 6.1. Secure system administration**

## 6.1 SSH Tectia Client Privileged User

The configuration of SSH Tectia Server (A) typically sets limitations on secure system administration. SSH Tectia Server (A) often resides in the DMZ. Strong two-factor authentication is often required from privileged users and connections are allowed only from certain hosts.

### 6.1.1 Disabling Root Login

Restrictions on Secure Shell services, as described for non-privileged users in Section 7.1.2 and Section 8.1.2, do not prevent users with shell access to the system from setting up the equivalent services.

It is also possible to limit users with administrative privileges to predefined commands if shell access is not needed.

Shell access is often desired for remote administration of the server computer. It is recommended to have users log in first to their non-privileged user accounts and once logged in elevate their rights using `sudo` or `su` especially if the `root` account is used instead of individual administrator accounts.

The following configuration setup prevents logging directly in to the privileged accounts:

```
<authentication-methods>
  <authentication action="deny">
    <selector>
      <user-privileged value="yes" />
    </selector>
  </authentication>
  ...
</authentication-methods>
```

### 6.1.2 Restricting Connections

SSH Tectia Server can be configured to reject connection attempts from unknown hosts. For example the following allows connections only from the internal network `10.1.0.0/8` IP addresses and from an external host with the IP address `195.20.116.1`:

```
<connections>
  <connection action="allow">
    <selector>
      <ip address="10.1.0.0/8" />
      <ip address="195.20.116.1" />
    </selector>
  </connection>
  <connection action="deny" />
</connections>
```

Please see the section called “Selectors” for information on the selectors.

On systems with several network interfaces, SSH Tectia Server can also be bound to a specific network interface so that the server can be only accessed from the intended network. For example, the following will bind the listener to address `10.1.60.25` using the Secure Shell default port 22:

```
<params>
  <listener id="intranet" address="10.1.60.25" />
  ...
</params>
```

### 6.1.3 Forced Commands

If you have maintenance jobs requiring non-interactive access to your server, use public-key authentication and forced commands. This way, if the private key is compromised, the public key cannot be used to perform anything other than the predetermined command on the server. (This is, of course, also bad, but it would be worse if the malicious attacker would have unrestricted access to the machine.)

Do not use the root account for jobs where it is not absolutely necessary.

You can set up a forced command in the `ssh-server-config.xml` file.

```
<services>
  <rule group="backup">
```

```
    <terminal action="deny" />
    <!-- This account is only used to backup the disk drive. -->
    <command application="dd if=/dev/hda" action="forced" />
    <tunnel-local action="deny" />
    <tunnel-remote action="deny" />
  </rule>
  ...
</services>
```

This would, on a successful login as the group `backup`, force a backup job to start.

You can also use the command that was given on the `sshg3` command line:

```
<services>
  <rule group="admin">
    <command application="echo $SSH2_ORIGINAL_COMMAND" action="forced" />
    ...
  </rule>
  ...
</services>
```

Running `sshg3`:

```

% sshg3 localhost kukkuu
kukkuu
%
```

## 6.2 Auditing

SSH Tectia Server logs events in the `syslog` on Unix and in the Windows Event Log on Windows. Logging (auditing) is very important for security. You should check your logs often, or use tools to analyze them. From the logs, you can see, for example, whether unauthorized access has been attempted, and take further action if needed. For example, you could set the hosts from which the attempts have been made as denied, or drop the packets from the domain completely at your firewall. The logs also provide troubleshooting information.

The log events are classified in seven levels, in decreasing order of importance:

**Security failure (Windows only)**
  A user tried to log on but failed.

**Security success (Windows only)**
  A user logged successfully on.

**Critical (Unix only)**
  A critical problem has occurred. By default, this is not used by SSH Tectia Server.

**Error**

A serious problem has occurred, preventing the intended operation from completing successfully.

**Warning**

A problem has occurred, but the operation can continue.

**Notice (Unix only)**

An action has been done.

**Informational**

Extra troubleshooting information.

## 6.2.1 Notification

It is recommended to notify the users before they decide to log in that their actions are logged. In some jurisdictions this is required.

To display, for example, the following text to the users before login, you can define a `banner-message` element in the `ssh-server-config.xml` file. See the section called "The `authentication-methods` Element".

```
Unauthorized use of this system is prohibited.
All actions are logged.
```

## 6.2.2 Customizing Logging

SSH Tectia Server allows customizing the severity and facility of different logging events. The events have reasonable default values, which are used if no explicit logging settings are made.

The logging settings are made in the `logging` element of the `ssh-server-config.xml` file. On Windows, the settings can be made with the **SSH Tectia Server Configuration** tool. See the section called "The `params` Element" and Section 4.1.5 for more information.

The default logging setting of SSH Tectia Server in the `ssh-server-config.xml` file is shown below:

```
<logging>
  <log-events facility="auth" severity="informational">
    Auth_method_success Auth_method_failure Auth_methods_completed
    Auth_methods_available Hostbased_auth_warning
    Publickey_auth_warning Publickey_auth_success GSSAPI_auth_warning
    Keyboard_interactive_pam_auth_warning
    Keyboard_interactive_radius_auth_warning
    Keyboard_interactive_password_auth_warning
    Keyboard_interactive_securid_auth_warning
    GSSAPI_auth_success
    Keyboard_interactive_pam_auth_success
    Keyboard_interactive_radius_auth_success
    Keyboard_interactive_password_auth_success
    Keyboard_interactive_securid_auth_success
  </log-events>
  <log-events facility="auth" severity="warning">
    Hostbased_auth_error Publickey_auth_error GSSAPI_auth_error
    Keyboard_interactive_pam_auth_error
    Keyboard_interactive_radius_auth_error
    Keyboard_interactive_password_auth_error
    Keyboard_interactive_securid_auth_error
  </log-events>
  <log-events facility="daemon" severity="error">
    Server_start_failed
  </log-events>
  <log-events facility="daemon" severity="warning">
    Server_listener_failed
  </log-events>
  <log-events facility="daemon" severity="notice">
    Server_listener_started
    Server_listener_stopped Server_reconfig_finished
    Server_stopping Server_running
    Server_starting
  </log-events>
  <log-events facility="daemon" severity="warning">
    Servant_exited Servant_error
  </log-events>
  <log-events facility="daemon" severity="informational">
    Server_reconfig_started
  </log-events>
  <log-events facility="normal" severity="informational">
    Algorithm_negotiation_success Certificate_validation_success
    Certificate_validation_failure Key_store_create
    Key_store_destroy Key_store_add_provider Key_store_decrypt
    Key_store_sign Key_store_sign_digest Logout Disconnect
    Channel_open_failure Session_channel_open
    Session_channel_close Forwarding_channel_open
    Forwarding_channel_open Forwarding_channel_close
    Forwarding_listener_open Forwarding_listener_close
    Auth_listener_open Auth_listener_close Auth_channel_open
    Auth_channel_close
  </log-events>
  <log-events facility="normal" severity="security-failure">
    Connection_denied Login_failure
  </log-events>
  <log-events facility="normal" severity="security-success">
    Connect Login_success
  </log-events>
  <log-events facility="normal" severity="warning">
    Algorithm_negotiation_failure KEX_failure
    Key_store_create_failed Key_store_add_provider_failed
    Key_store_decrypt_failed Key_store_sign_failed
    Key_store_sign_digest_failed
  </log-events>
</logging>
```

# Chapter 7 File Transfer

All versions of SSH Tectia Client and SSH Tectia Server provide the secure file transfer functionality. In addition to that, SSH Tectia Client (F) and SSH Tectia Server (F) provide advanced file transfer functionality, such as checkpoint/restart for the transfer of very large files, streaming for high-speed file transfers, and C and Java APIs for customization.

This chapter gives the typical SSH Tectia Server settings when it is used for secure file transfer and gives an example of an SSH Tectia Server (F) configuration.

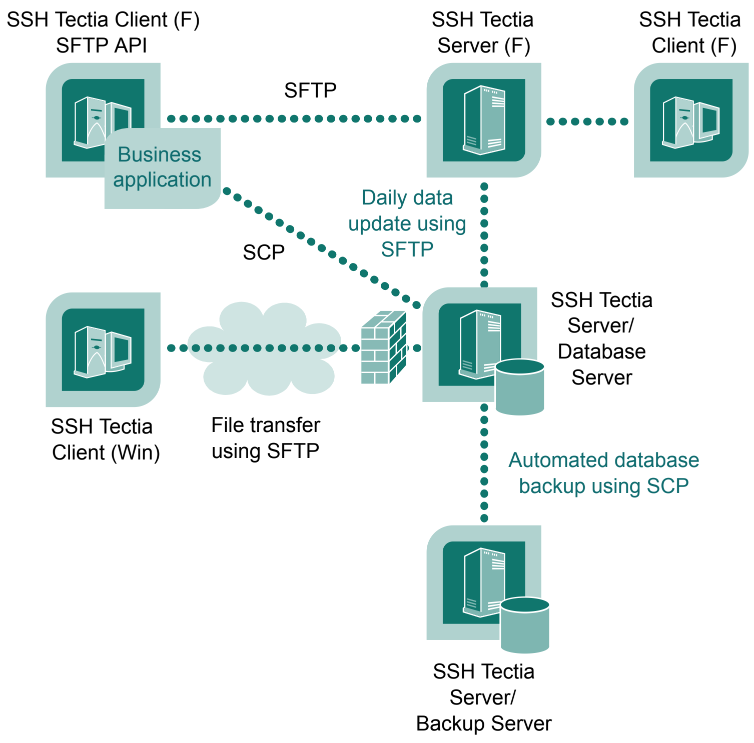


**Figure 7.1. Secure file transfer**

For more information on the advanced file transfer features available with SSH Tectia Server (F) and SSH Tectia Client (F), see *SSH Tectia Client/Server Product Description* and the documentation for the C and Java APIs (in the CD-ROM).

## 7.1 SSH Tectia Client File Transfer User

When SSH Tectia Server is used for automated file transfer, separate user accounts can be created for the file transfer users. Non-interactive authentication with public keys and scripted commands can be set for these accounts.

### 7.1.1 Encryption and Authentication Methods

The CryptiCore algorithm and public-key authentication should be enabled on the server.

#### Enabling CryptiCore

The CryptiCore algorithm is supported on x86-based processor architectures with SSH Tectia Server (F) and Server (T). It allows increased file transfer speeds for large file transfers.

To use CryptiCore, include the following in the `ssh-server-config.xml` file:

```
<connections>
  <connection action="allow" tcp-keepalive="no">
    <rekey seconds="3600" bytes="1000000000" />
    <cipher name="crypticore128@ssh.com" />
    <mac name="crypticore-mac@ssh.com" />
  </connection>
</connections>
```

#### Enabling Public-Key Authentication

To enable public-key authentication on the server, include the following in the `ssh-server-config.xml` file:

```
<authentication-methods login-grace-time="600">
  <banner-message />
  <auth-file-modes strict="yes" mask-bits="022" />
  <authentication>
    <auth-publickey />
  </authentication>
</authentication-methods>
```

The `auth-file-modes` element should be set to strict. This specifies that SSH Tectia Server on Unix checks the permissions and ownership of the user's key files used for public-key authentication.

## 7.1.2 Restricting Services

If SSH Tectia Server is used for file transfer only, it is typically sensible to disable tunneling and terminal access to the server.

### Enabling the SFT Subsystem

The secure file transfer subsystem can be defined in the `ssh-server-config.xml` file:

```
<services>
  <rule>
    <subsystem type="sftp" application="sft-server-g3" chroot="%homedir%" />
   ...
  </rule>
  ...
</services>
```

### Disabling Tunneling

If you are sure you or your users do not need to create tunnels (possibly going around firewall restrictions or such), you can disable tunneling (port forwarding) altogether by adding the following to the `ssh-server-config.xml` file:

```
<services>
  <rule>
   <tunnel-local action="deny" />
   <tunnel-remote action="deny" />
   ...
  </rule>
  ...
</services>
```

If you need more fine-grained control, you can define user groups in the `services` block and apply the restrictions only to the specified groups.

Tunneling restrictions can be further defined with the `src`, `dst`, and `listen` elements. See Chapter 8 for more information.

### Disabling Terminal Access

If you only want to enable file transfers or tunneling for users in group `remote-access`, you can disable terminal access by adding the following to the `ssh-server-config.xml` file:

```
<services>
  <rule group="remote-access">
    <terminal action="deny" />
    ...
  </rule>
  ...
</services>
```

This setting denies also X11 and agent forwarding and shell commands for the specified group (unless some commands are explicitly allowed).

The users will be able to use SFTP and other subsystems defined in the SSH Tectia Server configuration. Any other "exec" and "shell" requests will be denied for the users. This includes forced commands with public keys described in Section 6.1.3 and the legacy style password changing when performed as forced command.

### Defining Virtual Directories on Windows

If virtual folders are not defined, `HOME` and drive letters are use as defaults. If the `HOME` attribute is defined, its value is used, otherwise `%USERPROFILE%` is used in `HOME`.

If any virtual folders are defined, none of the defaults are used (drive letters, `HOME`). If you still want to use `HOME` and the drive letters, they need to be defined as virtual folders.

Virtual directories for SFTP can be defined as follows:

```
<subsystem type="sftp" application="sft-server-g3" action="allow">
 <!-- Virtual folders, Windows specific. -->
 <!-- These implicit default virtual folders are only set if no
      virtual folders are set in the configuration. If you set
      ANY virtual folders, none of the following will be set. -->
 <attribute name="home" value="%USERPROFILE%" />
 <attribute name="virtual-folder" value="C=C:/"/>
 <attribute name="virtual-folder" value="D=D:/"/>
 <attribute name="virtual-folder" value="E=E:/"/>
 <!-- ... all available drives. -->
      </subsystem>
```

## 7.1.3 Settings on the Client Side

For example, the following configuration can be used in the `ssh-broker-config.xml` file:

```
<profile name="sftexa"
              id="id1"
              host="sftexa.ssh.com"
              port="12345"
              connect-on-startup="no"
              user="sftuser">
       <ciphers>
         <cipher name="crypticore128@ssh.com" />
       </ciphers>
       <macs>
         <mac name="crypticore-mac@ssh.com" />
       </macs>
       <transport-distribution num-transports="4">
       </transport-distribution>
       <authentication-methods>
         <authentication-method name="publickey" />
       </authentication-methods>
       <compression name="none"/>
       <server-banners visible="no" />
</profile>
```

To enable non-interactive authentication, the private key on the Client is stored with a NULL passphrase. It is important that the key directory and the key file have the correct permissions (for example, 700). For more information, see Section 5.5.

## 7.2 Automated File Transfer

This section gives an example of setting up automated file transfer between SSH Tectia Client and Server hosts using scripts.

The following example script first transfers a file from SSH Tectia Client to SSH Tectia Server and then transfers the file back. The script logs the command and the return values to a file.

```
#!/bin/bash

DATE=`date +%d.%m.%Y-%H.%M`
SRV=sftexa

#scpg3 put
echo "/opt/tectia/bin/scpg3 -B -q testfile $SRV:test" >> scpg3_put_$DATE
/opt/tectia/bin/scpg3 -B -q testfile.dat $SRV:test
echo $? >> scpg3_put_$DATE

#scpg3 get
echo "/opt/tectia/bin/scpg3 -B -q $SRV:test test" >> scpg3_get_$DATE
/opt/tectia/bin/scpg3 -B -q $SRV:test test
echo $? >> scpg3_get_$DATE
```

The script can be set to run as a forced command. See Section 6.1.3.

# Chapter 8 Tunneling

Tunneling is a way to forward otherwise unsecured TCP traffic through Secure Shell. Tunneling can provide secure application connectivity, for example, to POP3-, SMTP-, and HTTP-based applications that would otherwise be unsecured.

The Secure Shell v2 connection protocol provides channels that can be used for a wide range of purposes. All of these channels are multiplexed into a single encrypted tunnel and can be used for tunneling (forwarding) arbitrary TCP/IP ports and X11 connections.

The client-server applications using the tunnel will carry out their own authentication procedures, if any, the same way they would without the encrypted tunnel.

The protocol/application might only be able to connect to a fixed port number (e.g. IMAP 143). Otherwise any available port can be chosen for tunneling. For remote (incoming) tunnels, the ports under 1024 (the well-known service ports) are not allowed for the regular users, but are available only for system administrators (root privileges).

There are two basic kinds of tunnels: local and remote. They are also called outgoing and incoming tunnels, respectively. X11 forwarding and agent forwarding are special cases of a remote tunnel.

SSH Tectia Client and all versions of SSH Tectia Server provide the basic tunneling functionality. SSH Tectia Connector and SSH Tectia Server (T) used together provide dynamic secure application tunneling that is transparent to the end user (secure application connectivity).

This chapter gives an example of SSH Tectia Server (T) settings for an SSH Tectia Connector tunneling user and describes the different tunneling options available with SSH Tectia Client and SSH Tectia Server.

## 8.1 SSH Tectia Connector Tunneling User

SSH Tectia Connector will connect only to SSH Tectia Server (T) or Server (M).

The SSH Tectia Connector users must be able to log in to an existing user account, preferably a non-privileged user account, on the server.

Users can have their own user accounts. If the Windows login name can be used also as the server-side login name, the variable `%USERNAME%` can be conveniently used in the configuration of SSH Tectia Connector.

Most of the authentication methods supported by SSH Tectia Server can be used with SSH Tectia Connector users. The authentication methods include password, any keyboard-interactive methods such as SecurID or RADIUS, public-key authentication with certificates on smart cards, and GSSAPI if the SSH Tectia Server and SSH Tectia Connector computers are part of the same Windows domain, or SSH Tectia Server can perform initial login to MIT Kerberos realm on behalf of the SSH Tectia Connector user.

User interaction is required for the keyboard-interactive authentication methods and typically at least the first time when the private key stored on a smart card is accessed in public-key authentication. Please see Chapter 5 for details of the user authentication methods.

### 8.1.1 Using a Shared Account

In case the tunneled applications provide sufficient user authentication, it is possible to use a shared user account, for example with a shared password, not requiring user interaction. Note that the shared account and password must only be used for tunneling, as the account is common to several users and the shared password is stored as plaintext in the SSH Tectia Connector configuration file.

See the operating system documentation for instructions on how to create a new user account, for example `tunnel`, with minimal privileges. It is very important that the shared user account is properly configured on the operating-system level.

The user should be denied at least shell access and the file system permissions should be restricted. This is done as a precaution in case the user is able to access the system using some other means than Secure Shell.

To deny shell access on the operating-system level, you can set the user's shell to `/bin/false` or use a script that can also inform the user of the situation:

For example, you could have the following saved to name `/bin/no-shell`:

```
#!/bin/sh

echo "Shell access to this account has been disabled."
exit 1
```

### 8.1.2 Restricting Services

In this example, the user `tunnel` is restricted to tunneling services while other users have terminal access. All users are denied file transfer service and X11 and agent forwarding.

Note that the users with terminal (shell) access are restricted only in the SSH Tectia Server configuration and can, for example, set up their own port forwardings. Please see Section 6.1 for more information.

## Tunneling

SSH Tectia Connector will use only outgoing tunnels. The tunnels are established based on the configuration of the application being tunneled. Please see Section 8.2 for details on the tunneling principles.

The following configuration options of SSH Tectia Server will deny incoming tunnels (remote port forwarding) and allow outgoing tunnels (local port forwarding) for all users for example to `http://webserver.example.com` or `https://webserver.example.com`.

```
<services>
  <rule>
    <tunnel-local action="allow">
      <dst fqdn="*.example.com" port="80" />
      <dst fqdn="*.example.com" port="443" />
    </tunnel-local>
    <tunnel-local action="deny" />
    <tunnel-remote action="deny" />
    ...
  </rule>
</services>
```

Note that the `fqdn` pattern does not match if the tunneled application uses IP addresses instead of DNS names for connections. The `address` pattern will match to both. xxx???

## Disabling Terminal Access

The following configuration options of SSH Tectia Server will deny the user `tunnel` terminal access.

```
<services>
  <group name="tunnel">
    <selector>
      <user name="tunnel" />
    </selector>
  </group>
  <rule group="tunnel">
    <terminal action="deny" />
    <command action="forced" application="no-shell" />
    ...
  </rule>
  ...
</services>
```

Denying terminal denies also X11 and agent forwarding and shell commands (unless some commands are explicitly allowed).

The `command` action in this example, provides an alternative method of informing the user of denied shell access using the `/bin/no-shell` script introduced in Section 8.1.1).

This method can be used if the risk of gaining access via other means than Secure Shell can be eliminated. This way, each user's shell does not have to be set separately, and the setting can be easily scaled to several users.

### Disabling File Transfers

To deny all users the access to the SFTP server, change the default SFTP subsystem configuration option of SSH Tectia Server to:

```
...
  <rule>
    ...
    <subsystem type="sftp" action="deny" />
     ...
  </rule>
  ...
```

## 8.2 Local Tunnels

A local (outgoing) tunnel forwards traffic coming to a local port to a specified remote port.

Setting up local tunneling allocates a listener port on the local client. Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the remote server and another connection is made from the server to a specified host and port. The connection from the server onwards will not be secure as it is a normal TCP connection.

For example, when you use SSH Tectia Client on the command line, and issue the following command, all traffic coming to port 1234 on the client will be forwarded to port 23 on the server. See Figure 8.1.

```
$ sshg3 -L 1234:localhost:23 username@sshserver
```

The forwarding address in the command is resolved at the (remote) end point of the tunnel. In this case `localhost` refers to the server host (`sshserver`).

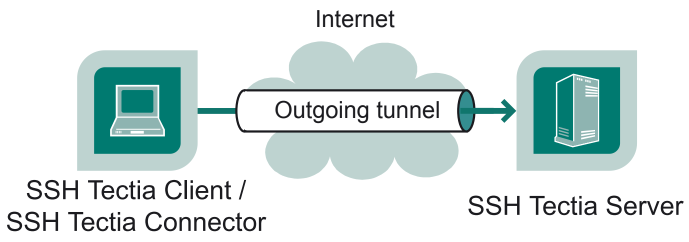


**Figure 8.1. Simple local (outgoing) tunnel**

To use the tunnel, the application to be tunneled is set to connect to the local listener port instead of connecting to the server directly. SSH Tectia Client forwards the connection securely to the remote server.

If you have three hosts, for example, `sshclient`, `sshserver`, and `imapserver`, and you forward the traffic coming to the `sshclient` port `143` to the `imapserver` port `143`, only the connection between `sshclient` and `sshserver` will be secured. The command you use would be similar to the following:

```
$ sshg3 -L 143:imapserver:143 username@sshserver
```

Figure 8.2 shows an example where the Secure Shell server resides in the DMZ network. The connection is encrypted from the Secure Shell client to the Secure Shell server and continues unencrypted in the corporate network to the IMAP server.

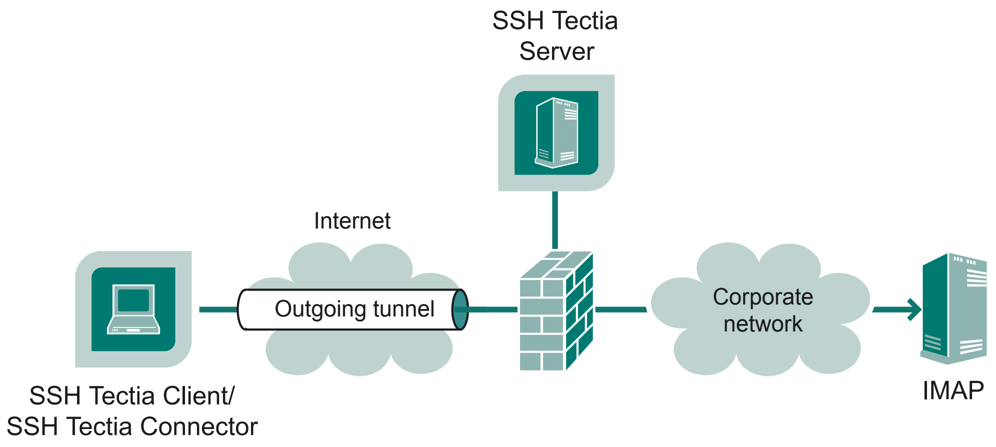


**Figure 8.2. Local (outgoing) tunnel to an IMAP server**

With SSH Tectia Connector, there is no need to separately configure application software to use local ports to set up the tunnels. The applications to be tunneled are defined in the Connection Broker configuration (**Filter Rules**). SSH Tectia Connector automatically captures the defined applications and the Connection Broker creates Secure Shell tunnels to the defined SSH Tectia Server (T).

By default, local tunnels are allowed to all addresses for all users. The default setting equals the following in the `ssh-server-config.xml` file:

```
<services>
  <rule>
    <tunnel-local action="allow" />
    ...
  </rule>
</services>
```

The connections can be restricted by specifying allowed addresses with the `src` and `dst` elements.

If any addresses are specified as allowed, local tunnels to all other addresses are implicitly denied.

For example, the following setting allows tunneling to the e-mail server, but denies all other tunnels:

```
<services>
  <rule>
```

```
    <tunnel-local action="allow">
    <!-- IMAP. -->
    <dst fqdn="imap.example.com" port="143" />
    <dst fqdn="imap.example.com" port="993" />
    <!-- POP. -->
    <dst fqdn="imap.example.com" port="109" />
    <dst fqdn="imap.example.com" port="110" />
    <dst fqdn="imap.example.com" port="995" />
  </tunnel-local>
    ...
  </rule>
</services>
```

In the **SSH Tectia Server Configuration** GUI on Windows, tunneling settings are made on the **Services** page under **Rules**. See Section 4.1.10.

## 8.3 Remote Tunnels

A remote (incoming) tunnel it forwards traffic coming to a remote port to a specified local port.

Setting up remote tunneling allocates a listener port on the remote server. Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the local client and another connection is made from the client to a specified host and port. The connection from the client onwards will not be secure as it is a normal TCP connection.

For example, if you issue the following command, all traffic which comes to port 1234 on the server will be forwarded to port 23 on the client. See Figure 8.3.

```
$ sshg3 -R 1234:localhost:23 username@sshserver
```

The forwarding address in the command is resolved at the (local) end point of the tunnel. In this case `localhost` refers to the client host.

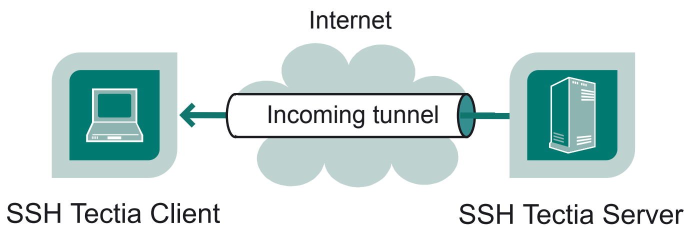


**Figure 8.3. Remote (incoming) tunnel**

By default, remote tunnels are allowed from all addresses for all users. The default setting equals the following in the `ssh-server-config.xml` file:

```
<services>
  <rule>
```

```
    <tunnel-remote action="allow" />
    ...
  </rule>
</services>
```

The connections can be restricted by specifying allowed addresses with the `src` and `listen` elements.

If any addresses are specified as allowed, remote tunnels to all other addresses are implicitly denied.

In the **SSH Tectia Server Configuration** GUI on Windows, tunneling settings are made on the **Services** page under **Rules**. See Section 4.1.10.

## 8.4 X11 Forwarding (Unix)

X11 forwarding is a special case of remote tunneling.

SSH Tectia Server supports X11 forwarding on Unix platforms. SSH Tectia Client supports X11 forwarding on both Unix and Windows platforms.

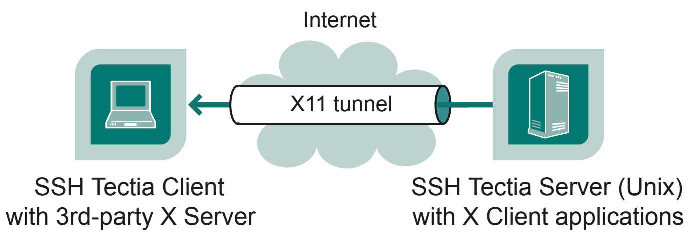


**Figure 8.4. X11 forwarding**

By default, X11 forwarding is enabled for all users. To enable X11 forwarding only for the specified users, include an entry similar to the following in your `ssh-server-config.xml` file:

```
<services>
  <rule group="admins">
    <tunnel-x11 action="allow" />
    ...
  </rule>
  <rule>
    <tunnel-x11 action="deny" />
  </rule>
</services>
```

## 8.5 Agent Forwarding (Unix)

Agent forwarding is a special case of remote tunneling. In agent forwarding, Secure Shell connections and public-key authentication data are forwarded from one server to another without the user having to authenticate separately for each server. Authentication data does not have to be stored on any other machine than the local machine, and authentication passphrases or private keys never go over the network.

SSH Tectia Client and Server support agent forwarding on Unix platforms. The servers in the middle of the forwarding chain must have also a Secure Shell client component installed.

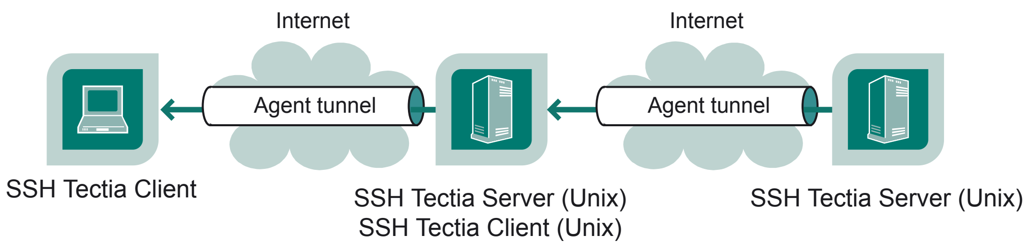


**Figure 8.5. Agent forwarding**

By default, agent forwarding is enabled for all users. To enable agent forwarding only for the specified users, include an entry similar to the following in your `ssh-server-config.xml` file:

```
<services>
  <rule group="admins">
    <tunnel-agent action="allow" />
    ...
  </rule>
  <rule>
    <tunnel-agent action="deny" />
  </rule>
</services>
```

# Appendix A Command-Line Tools

The functionality of the command-line tools included in SSH Tectia Server is briefly explained in the following appendices.

## ssh-server-g3

ssh-server-g3 -- Secure Shell server - Generation 3

### Synopsis

`ssh-server-g3` [`-D, --debug=`*LEVEL*] [`-f, --config-file=`*FILE*] [`-h, --help`] [`-H, --hostkey=`*FILE*] [`-l, --listen=`[*ADDRESS:*]*PORT*] [`-n, --num-processes=`*NUM*] [`-V, --version`] [`--plugin-path=`*PATH*] [`--auxdata-path=`*PATH*] [`--libexec-path=`*PATH*] [`--allowed-ciphers=`*LIST*] [`--allowed-macs=`*LIST*] [`--fips-mode` [`={yes|no}`]]

### Description

`ssh-server-g3` is the Secure Shell server program for SSH Tectia Server.

The `ssh-server-g3` command should not be used directly, except for debugging purposes. Use instead the startup script with the same name, `ssh-server-g3`.

The path to the `ssh-server-g3` startup script is different on each operating system:

- On AIX:

  ```
  # /opt/tectia/sbin/rc.ssh-server-g3 [command]
  ```

- On Linux and Solaris:

  ```
  # /etc/init.d/ssh-server-g3 [command]
  ```

- On HP-UX:

  ```
  # /sbin/init.d/ssh-server-g3 [command]
  ```

The `command` can be either `start`, `stop`, `restart`, or `reload`.

**start**

Start the server.

**stop**

Stop the server. Existing connections stay open until closed from the client side.

**restart**

Start a new server process. Existing connections stay open using the old server process. The old process is closed after the last old connection is closed from the client side.

**reload**

Reload the configuration file. Existing connections stay open.

## Options

When the `ssh-server-g3` command is used directly, it accepts the following options:

`-D, --debug=`*`LEVEL`*

Sets the debug level.

`-f, --config-file=`*`FILE`*

Specifies the configuration file to be used.

`-h, --help`

Prints a short summary of command-line options and exits.

`-H, --hostkey=`*`FILE`*

Specifies the host key file to be used.

`-l, --listen=` [*`ADDRESS:`*]*`PORT`*

Specifies the listen address and port. If *`ADDRESS`* is unspecified, listen on any IP address.

`-n, --num-processes=`*`NUM`*

Sets the initial number of Servant processes.

`-V, --version`

Prints program version string and exits.

`--plugin-path=`*`PATH`*

Sets the path to the plugin directory.

`--auxdata-path=`*`PATH`*

Sets the path to the auxiliary data directory.

`--libexec-path=`*`PATH`*

Sets the path to the `libexec` directory.

`--allowed-ciphers=`*`LIST`*

Sets a list of allowed ciphers.

`--allowed-macs=`*`LIST`*

Sets a list of allowed MACs.

`--fips-mode [={yes|no}]`

Sets the FIPS mode for the cryptographic library in use (or not in use). The default is *`no`*.

## Authors

SSH Communications Security Corp.

For more information, see http://www.ssh.com.

## See Also

ssh-server-config-tool(8), ssh-server-config(5)

# ssh-server-config-tool

ssh-server-config-tool -- SSH Tectia Server configuration tool

## Synopsis

`ssh-server-config-tool [-D, --debug=`*`LEVEL`*`] [-h, --help] [-p, --port] [-V, --version]`

## Description

`ssh-server-config-tool` is a tool for initiating reconfiguration of SSH Tectia Server (`ssh-server-g3`). It reads the configuration from the `ssh-server-config.xml` file. Existing connection stay open using the old configuration and the new connections will use the new configuration.

`ssh-server-config-tool` returns `0` on success and a positive integer on failure.

## Options

`-D, --debug=`*`LEVEL`*

Sets the debug level.

`-h, --help`
: Prints a short summary of command-line options and exits.

`-p, --port`
: Specifies the port number of the server process being controlled.

`-V, --version`
: Prints program version string and exits.

### Authors

SSH Communications Security Corp.

For more information, see http://www.ssh.com.

### See Also

ssh-server-g3(8), ssh-server-config(5)

## ssh-keygen-g3

ssh-keygen-g3 -- authentication key pair generator

### Synopsis

`ssh-keygen-g3` [`options`...]
[*`key1 key2`*...]

### Description

`ssh-keygen-g3` (`ssh-keygen-g3.exe` on Windows) is a tool that generates and manages authentication keys for Secure Shell. Each user wishing to use a Secure Shell client with public-key authentication can run this tool to create authentication keys. Additionally, the system administrator can use this to generate host keys for the Secure Shell server.

By default, if no path for the key files is specified, the key pair is generated under the user's home directory (`$HOME/.ssh2` on Unix, `%USERPROFILE%\Application Data\SSH\UserKeys` on Windows). If no filename is specified, the key pair is likewise stored under the user's home directory with such filenames as `id_dsa_1024_a` and `id_dsa_1024_a.pub`.

### Options

The following options are available:

`-b` *`bits`*

Specifies the length of the key in bits (default *`2048`*).

`-t` *`dsa`*|*`rsa`*

Selects the type of the key. Valid options are *`dsa`* (default) and *`rsa`*.

`--fips-mode` [`={yes|no}`]

Generates the key using the FIPS mode for the cryptographic library. The default is *`no`*.

`--fips-crypto-dll-path` *`path`*

Specifies the location of the FIPS cryptographic DLL.

`-c` *`comment_string`*

Specifies the key's comment string.

`-e` *`file`*

Edits the specified key. Makes `ssh-keygen-g3` interactive. You can change the key's passphrase or comment.

`-p` *`passphrase`*

Specifies the passphrase used.

`-P`

Specifies that the key will be saved with an empty passphrase.

`-h | -?`

Displays help and exits.

`-q`

Hides the progress indicator.

`-1` *`file`*

Converts a key from the SSH1 format to the SSH2 format.

`-i` *`file`*

Loads and displays information on *`file`*.

`-D` *`file`*

Derives the public key from the private key *`file`*.

`-B` *`number`*

Specifies the number base for displaying key information (default *`10`*).

`-V`

Displays version string and exits.

`-r` *`file`*

Adds entropy from *`file`* to the random pool. If *`file`* contains 'relatively random' data (i.e. data unpredictable by a potential attacker), the randomness of the pool is increased. Good randomness is essential for the security of the generated keys.

`--overwrite [={yes|no}]`

Overwrite files with the same filenames. The default is to overwrite.

`-x` *`file`*

Converts a private key from the X.509 format to the SSH2 format.

`-k` *`file`*

Converts a PKCS #12 file to an SSH2-format certificate and private key.

`-7` *`file`*

Extracts certificates from a PKCS #7 file.

`-F` *`file`*

Dumps the fingerprint of the given public key. The fingerprint is given in the Bubble Babble format, which makes the fingerprint look like a string of "real" words (making it easier to pronounce).

`-H, --hostkey`

Generates a Secure Shell host key pair and stores the key pair in the default host key directory (`/etc/ssh2` on Unix, `C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Server` on Windows).

`--import-public-key` *`infile outfile`*

Attempts to import a public key from *`infile`* and store it to *`outfile`* in SSH2 native format.

`--import-private-key` *`infile outfile`*

Attempts to import an unencrypted private key from *`infile`* and store it to *`outfile`* in SSH2 native private key format.

`--import-ssh1-authorized-keys` *`infile outfile`*

Imports an SSH1-style authorized_keys file *`infile`* and generates an SSH2-style authorization file *`outfile`* and stores the keys from *`infile`* to generated files into the same directory with *`outfile`*.

## Authors

SSH Communications Security Corp.

For more information, see http://www.ssh.com.

## See Also

sshg3(1)

# ssh-certview-g3

ssh-certview-g3 -- certificate viewer

## Synopsis

`ssh-certview-g3`
[`options`...] *`file`*
[`options`...] *`file ...`*

## Description

The `ssh-certview-g3` program (`ssh-certview-g3.exe` on Windows) is a simple command-line application, capable of decoding and showing X.509 certificates, CRLs, and certification requests. The command output is written to the standard output.

## Options

The following options are available:

`-h`
: Displays a short help.

`-verbose`
: Gives more diagnostic output.

`-quiet`
: Gives no diagnostic output.

`-auto`
: The next input file type is auto-detected (default).

`-cert`
: The next input file is a certificate.

`-certpair`
: The next input file is a cross-certificate pair.

`-crmf`
: The next input file is a CRMF certification request.

`-req`
: The next input file is a PKCS #10 certification request.

`-crl`
: The next input file is a CRL.

`-prv`
: The next input file is a private key.

`-pkcs12`
: The next input file is a PKCS#12 package.

`-ssh2`
: The next input file is an SSH2 public key.

`-spkac`
: The next input file is a Netscape-generated SPKAC request.

`-noverify`
: Does not check the validity of the signature on the input certificate.

`-autoenc`
: Determines PEM/DER automatically (default).

`-pem`

Assumes that the input file is in PEM (ASCII base-64) format. This option allows both actual PEM (with headers and footers), and plain base-64 (without headers and footers). An example of PEM header and footer is shown below:

```
-----BEGIN CERTIFICATE-----
encoded data
-----END CERTIFICATE-----
```

`-der`
: Assumes that the input file is in DER format.

`-hexl`

Assumes that the input file is in Hexl format. (Hexl is a common Unix tool for outputting binary files in a certain hexadecimal representation.)

`-skip` *`number`*
: Skips *`number`* bytes from the beginning of input before trying to decode. This is useful if the file contains some garbage before the actual contents.

`-ldap`
: Prints names in LDAP order.

`-utf8`
: Prints names in UTF-8.

`-latin1`
: Prints names in ISO-8859-1.

`-base10`
: Outputs big numbers in base-10 (default).

`-base16`
: Outputs big numbers in base-16.

`-base64`
: Outputs big numbers in base-64.

`-width` *`number`*
: Sets output width (*`number`* characters).

## Example

For example, using a certificate downloaded from `pki.ssh.com`, when the following command is given:

```
$ ssh-certview-g3 -width 70 ca-certificate.cer
```

The following output is produced:

```
Certificate =
  SubjectName = <C=FI, O=SSH Communications Security Corp, CN=Secure
    Shell Test CA>
  IssuerName = <C=FI, O=SSH Communications Security Corp, CN=Secure
    Shell Test CA>
  SerialNumber= 34679408
  SignatureAlgorithm = rsa-pkcs1-sha1
  Certificate seems to be self-signed.
      * Signature verification success.
  Validity =
    NotBefore = 2003 Dec  3rd, 08:04:27 GMT
    NotAfter  = 2005 Dec  2nd, 08:04:27 GMT
  PublicKeyInfo =
    PublicKey =
      Algorithm name (SSH) : if-modn{sign{rsa-pkcs1-md5}}
      Modulus n  (1024 bits) :
        9635680922805930263476549641957998756341022541202937865240553
        9374740946079473767424224071470837728840839320521621518323377
        3593102350415987252300817926769968881159896955490274368606664
        0759644131690750532665266218696466060377799358036735475902257
        6086098562919363963470926690162744258451983124575595926849551
        903
      Exponent e (  17 bits) :
        65537
  Extensions =
    Available = authority key identifier, subject key identifier, key
      usage(critical), basic constraints(critical), authority
      information access
    KeyUsage = DigitalSignature KeyEncipherment KeyCertSign CRLSign
        [CRITICAL]
    BasicConstraints =
      PathLength = 0
      cA         = TRUE
        [CRITICAL]
    AuthorityKeyID =
      KeyID =
        eb:f0:4d:b5:b2:4c:be:47:35:53:a8:37:d2:8d:c8:b2:f1:19:71:79
    SubjectKeyID =
      KeyId =
        eb:f0:4d:b5:b2:4c:be:47:35:53:a8:37:d2:8d:c8:b2:f1:19:71:79
    AuthorityInfoAccess =
      AccessMethod = 1.3.6.1.5.5.7.48.1
      AccessLocation =
        Following names detected =
          URI (uniform resource indicator)
        Viewing specific name types =
          URI = http://pki.ssh.com:8090/ocsp-1/
  Fingerprints =
    MD5 = c7:af:e5:3d:f6:ea:ce:da:07:93:d0:06:8d:c0:0a:f8
    SHA-1 =
    27:d7:19:47:7c:08:3e:1a:27:4b:68:8e:18:83:e8:f9:23:e8:29:85
```

## Authors

SSH Communications Security Corp.

For more information, see http://www.ssh.com.

# ssh-cmpclient-g3

ssh-cmpclient-g3 -- CMP enrollment client

## Synopsis

```
ssh-cmpclient-g3 command [options] access [name]

Where command is one of the following:

     INITIALIZE psk|racerts keypair template
     ENROLL certs|racerts keypair template
     UPDATE certs [keypair]
     POLL psk|certs|racerts id
     RECOVER psk|certs|racerts template
     REVOKE psk|certs|racerts template
     TUNNEL racerts template

Most commands can accept the following options:
     -B            Perform key backup for subject keys.
     -o prefix     Save result into files with prefix.
     -O filename   Save the result into the specified file.
                   If there is more than one result file,
                   the remaining results are rejected.
     -C file       CA certificate from this file.
     -S url        Use this SOCKS server to access the CA.
     -H url        Use this HTTP proxy to access the CA.
     -E            PoP by encryption (CA certificate needed).
     -v num        Protocol version 1|2 of the CA platform. Default is 2.
     -y            Non-interactive mode. All questions answered with 'y'.
     -N file       Specifies a file to stir to the random pool.

The following identifiers are used to specify options:
     psk      -p refnum:key (reference number and pre-shared key)
              -p file (containing refnum:key)
              -i number (iteration count, default 1024)
     certs    -c file (certificate file) -k url (private-key URL)
     racerts  -R file (RA certificate file) -k url (RA private-key URL)
     keypair  -P url (private-key URL)
     id       -I number (polling ID)
     template -T file (certificate template)
              -s subject-ldap[;type=value]
              -u key-usage-name[;key-usage-name]
              -U extended-key-usage-name[;extended-key-usage-name]
     access   URL where the CA listens for requests.
     name     LDAP name for the issuing CA (if -C is not given).

Key URLs are either valid external key paths or in the format:
     "generate://savetype:passphrase@keytype:size/save-file-prefix"
     "file://passphrase/absolute-key-file-path"
     "file:/absolute-key-file-path"
     "file:relative-key-file-path"
     "any-key-file-path"

The key generation "savetype" can be:
 - ssh2, secsh2, secsh (Secure Shell 2 key type)
 - ssh1, secsh1 (legacy Secure Shell 1 key type)
 - pkcs1 (PKCS #1 format)
 - pkcs8s (passphrase-protected PKCS #8, "shrouded PKCS #8")
 - pkcs8 (plain-text PKCS #8)
 - x509 (SSH-proprietary X.509 library key type)

     -h Prints usage message.
     -F Prints key usage extension and keytype instructions.
     -e Prints command-line examples.
```

## Description

The `ssh-cmpclient-g3` command-line tool (`ssh-cmpclient-g3.exe` on Windows) is a certificate enrollment client that uses the CMP protocol. It can generate an RSA or DSA public-key pair and get certificates for their public components. CMP is specified by the IETF PKIX Working Group for certificate life-cycle management, and is supported by some CA platforms, such as Entrust PKI and RSA Keon.

## Commands

The `ssh-cmpclient-g3` command-line command keywords are listed below. Shorthands longer than three letters can be used to identify the command. The commands are case-insensitive. The user must specify the CA address URL for each command. Here the term "user" refers to a user, program, or hardware device.

`INITIALIZE`
: Requests the user's initial certificate. The request is authenticated using the reference number and the corresponding key (PSK) received from the CA or RA using some out-of-band mechanism.

  The user must specify the PSK, the asymmetric key pair, and a subject name.

`ENROLL`
: Requests a new certificate when the user already has a valid certificate for the key. This request is similar to `initialize` except that it is authenticated using public-key methods.

`POLL`
: Polls for a certificate when a request was not immediately accepted.

`UPDATE`
: Requests an update of an existing certificate (replacement). The issued certificate will be similar to the existing certificate (names, flags, and other extensions). The user can change the key, and the validity times are updated by the CA. This request is authenticated by a valid existing key pair and a certificate.

`RECOVER`
: Requests recovery of a backed-up key. This request is authenticated either by PSK-based or certificate-based authentication. The template describes the certificate whose private key has already been backed up and should be recovered. Users can only recover keys they have backed up themselves.

`REVOKE`
: Requests revocation for a key specified in the template. Authentication of the request is made using a PSK or a certificate belonging to the same user as the subject of revocation.

`TUNNEL`

Operates in RA tunnel mode. Reads requests and optionally modifies the subject name, alternative names, and extensions based on the command line. Approves the request and sends it to the CA.

## Options

The `ssh-cmpclient-g3` command-line options are listed below. Note that when a file name is specified, an existing file with the same name will be overwritten. When subject names or other strings that contain spaces are given on the command line, they should be enclosed in double quotes.

`-B`

Requests private key backup to be performed for the initialize, enroll, and update commands.

`-o` *`prefix`*

Saves resulting certificates and CRLs into files with the given prefix. The prefix is first appended by a number, followed by the file extension `.crt` or `.crl`, depending on the type of object.

`-O` *`filename`*

Saves the result into the specified absolute filename. If there is more than one result file, the remaining results are rejected.

`-C` *`file`*

Specifies the file path that contains the CA certificate. If key backup is done, the file name must be given, but in most cases the LDAP name of the CA can be given instead.

`-S` *`url`*

Specifies the SOCKS URL if the CA is located behind a SOCKS- enabled firewall. The format of the URL is: `socks://[username@]server[:port][/network/bits[,network/bits]]`

`-H` *`url`*

Uses the given HTTP proxy server to access the CA. The format of the URL is: `http://server[:port]/`

`-E`

Performs encryption proof of possession if the CA supports it. In this method of PoP, the request is not signed, but instead the PoP is established based on the ability to decrypt the certificates received from the CA. The CA encrypts the certificates with the user's public key before sending them to the user.

`-v` *`num`*

Selects the CMP protocol version. This is either value 1, for an RFC 2510-based protocol, or 2 (the default) for CMPv2.

`-N` *`file`*

Specifies a file to be used as an entropy source during key generation.

The usage line uses the following meta commands:

`psk`

The reference number and the corresponding key value given by the CA or RA.

`-p` *`refnum:key|file`*

`refnum` and `key` are character strings shared among the CA and the user. `refnum` identifies the secret `key` used to authenticate the message. The `refnum` string must not contain colon characters.

Alternatively, a filename containing the reference number and the key can be given as the argument.

`-i` *`number`*

`number` indicates the key hashing iteration count.

`certs`

The user's existing key and certificate for authentication.

`-k` *`url`*

URL specifying the private key location. This is an external key URL whose format is specified in Section the section called “Synopsis”.

`-c` *`file`*

Path to the file that contains the certificate issued to the public key given in the `-k` option argument.

`racerts`

In RA mode, the RA key and certificate for authentication.

`-k` *`url`*

URL specifying the private key location. This is an external key URL whose format is specified in Section the section called “Synopsis”.

`-R` *`file`*

Path to the file that contains the RA certificate issued to the public key given in the `-k` option argument.

`keypair`

The subject key pair to be certified.

`-P` *`url`*

URL specifying the private key location. This is an external key URL whose format is specified in Section the section called “Synopsis”.

`id`

Polling ID used if the PKI action is left pending.

`-I number`

Polling transaction ID `number` given by the RA or CA if the action is left pending.

`template`

The subject name and flags to be certified.

`-T` *`file`*

The file containing the certificate used as the template for the operation. Values used to identify the subject are read from this, but the user can overwrite the key, key-usage flags, or subject names.

`-s` *`subject-ldap[;type=value]*`*

A subject name in reverse LDAP format, that is, the most general component first, and alternative subject names. The name `subject-ldap` will be copied into the request verbatim.

A typical choice would be a DN in the format `"C=US,O=SSH,CN=Some Body"`, but in principle this can be anything that is usable for the resulting certificate.

The possible `type` values are `ip`, `email`, `dn`, `dns`, `uri`, and `rid`.

`-u` *`key-usage-name[;key-usage-name]*`*

Requested key usage purpose code. The following codes are recognized: `digitalSignature`, `nonRepudiation`, `keyEncipherment`, `dataEncipherment`, `keyAgreement`, `keyCertSign`, `cRLSign`, `encipherOnly`, `decipherOnly`, and `help`. The special keyword help lists the supported key usages which are defined in RFC 3280.

`-U` *`extended-key-usage-name[;extended-key-usage-name]*`*

Requested extended key usage code. The following codes, in addition to user-specified dotted OID values are recognized: `serverAuth`, `clientAuth`, `codeSigning`, `emailProtection`, `timeStamping`, `ikeIntermediate`, and `smartCardLogon`.

`access`

Specifies the CA address in URL format. Possible access methods are HTTP (`http://host:port/path`), or plain TCP (`tcp://host:port/path`). If the host address is an IPv6 address, it must be enclosed in square brackets (`http://[IPv6-address]:port/`).

`name`

Optionally specifies the destination CA name for the operation, in case a CA certificate was not given using the option `-C`.

## Examples

### Initial Certificate Enrollment

This example provides commands for enrolling an initial certificate for digital signature use. It generates a private key into a PKCS #8 plaintext file named `initial.prv`, and stores the enrolled certificate into file `initial-0.crt`. The user is authenticated to the CA with the key identifier (refnum) `62154` and the key `ssh`. The subject name and alternative IP address are given, as well as key-usage flags. The CA address is `pki.ssh.com`, the port `8080`, and the CA name to access `Test CA 1`.

```
$ ssh-cmpclient-g3 INITIALIZE \
   -P generate://pkcs8@rsa:1024/initial -o initial \
   -p 62154:ssh \
   -s 'C=FI,O=SSH,CN=Example/initial;IP=1.2.3.4' \
   -u digitalsignature \
   http://pki.ssh.com:8080/pkix/ \
   'C=FI, O=SSH Communications Security Corp, CN=SSH Test CA 1 No Liabilities'
```

As a response the command presents the issued certificate to the user, and the user accepts it by typing `yes` at the prompt.

```
Certificate =
  SubjectName = <C=FI, O=SSH, CN=Example/initial>
  IssuerName = <C=FI, O=SSH Communications Security Corp,
    CN=SSH Test CA 1 No Liabilities>
  SerialNumber= 8017690
  SignatureAlgorithm = rsa-pkcs1-sha1
  Validity = ...
  PublicKeyInfo = ...
  Extensions =
      Viewing specific name types = IP = 1.2.3.4
    KeyUsage = DigitalSignature
    CRLDistributionPoints = ...
    AuthorityKeyID =
      KeyID = 3d:cb:be:20:64:49:16:1d:88:b7:98:67:93:f0:5d:42:81:2e:bd:0c
    SubjectKeyID =
      KeyId = 6c:f4:0e:ba:b9:ef:44:37:db:ad:1f:fc:46:e0:25:9f:c8:ce:cb:da
  Fingerprints =
    MD5 = b7:6d:5b:4d:e0:94:d1:1f:ec:ca:c2:ed:68:ac:bf:56
    SHA-1 = 4f:de:73:db:ff:e8:7d:42:c4:7d:e1:79:1f:20:43:71:2f:81:ff:fa

Do you accept the certificate above? yes
```

## Key update

Before the certificate expires, a new certificate with updated validity period should be enrolled. `ssh-cmpclient-g3` supports key update, where a new private key is generated and the key update request is authenticated with the old (still valid) certificate. The old certificate is also used as a template for issuing the new certificate, so the identity of the user will not be changed during the key update. With the following command you can update the key pair, which was enrolled in the previous example. Presenting the resulting certificate has been left out.

```
$ ssh-cmpclient-g3 UPDATE \
   -k initial.prv -c initial-0.crt -P \
   generate://pkcs8@rsa:1024/updatedcert -o updatedcert \
   http://pki.ssh.com:8080/pkix/ \
   "C=FI, O=SSH Communications Security Corp, CN=SSH Test CA 1 No Liabilities"
```

The new key pair can be found in the files with the `updatedcert` prefix. The policy of the issuing CA needs to also allow automatic key updates if `ssh-cmpclient-g3` is used in the `UPDATE` mode.

## Authors

SSH Communications Security Corp.

For more information, see http://www.ssh.com.

# ssh-ekview-g3

ssh-ekview-g3 -- external key viewer

## Synopsis

`ssh-ekview-g3` [`options`...] *`provider`*

## Description

The `ssh-ekview-g3` program (`ssh-ekview-g3.exe` on Windows) allows you to export certificates from external key providers such as Entrust. You can further study these certificates with `ssh-certview-g3`.

This is useful when you want to generate, for example, entries for allowing certificate authentication in the `ssh-server-config.xml` file. You might need to know the subject names on the certificate.

With `ssh-ekview-g3`, you can export the certificate and get the information you need from the certificates with `ssh-certview-g3`.

## Options

The following options are available:

`-h`

Displays a short help.

`-i` *`info`*

Uses *`info`* as the initialization string for the provider.

`-k`

Prints the key paths only.

`-e` *`keypath`*

Exports certificates at *`keypath`* to files.

`-a`

Exports all found certificates to files.

`-b` *`base`*

Uses *`base`* when printing integers. For example, the decimal 10 is 'a' in base-16.

`-d` *`debug_level`*

Prints extensive debug information to stderr. *`debug_level`* is either a number, from 0 to 99, where 99 specifies that all debug information should be displayed, or a comma-separated list of assignments of the format `ModulePattern=debug_level`, for example `"*=10,sshd2=2"`. This should be the first argument on the command line.

## Example

For example the following command will dump all certificates in the `entrust` provider to files:

```
ssh-ekview-g3 -a -i"ini-file($HOME/my.ini) profile-file($HOME/solo.ini)" entrust
```

## Authors

SSH Communications Security Corp.

For more information, see http://www.ssh.com.

## See Also

ssh-certview-g3(1), ssh-server-config(5)

# Appendix B Server Configuration File Syntax

The DTD of the server configuration file is shown below:

```
<!--         -->
<!--         -->
<!-- secsh-server.dtd       -->
<!--         -->
<!-- Author: Markku Rossi <mtr@ssh.com>     -->
<!--      Sami Lehtinen <sjl@ssh.com>    -->
<!--         -->
<!-- Copyright (c) 2004-2005 SSH Communications Security, Finland -->
<!--     All rights reserved.     -->
<!--         -->
<!-- Document type definition for SecSh server XML configuration -->
<!-- files.        -->
<!--         -->
<!--         -->

<!-- Tunable parameters used in the policy. -->

<!-- Default connection action. -->
<!ENTITY % default-connection-action   '"allow"'>

<!-- Default terminal action. -->
<!ENTITY % default-terminal-action   '"allow"'>

<!-- Default subsystem action. -->
<!ENTITY % default-subsystem-action   '"allow"'>

<!-- Default user-privileged value. -->
<!ENTITY % default-user-privileged-value  '"yes"'>

<!-- Default user-password-change-needed value. -->
<!ENTITY % default-user-password-change-needed-value '"yes"'>

<!-- Default tunnel action. -->
<!ENTITY % default-tunnel-action   '"allow"'>
```

```
<!-- Default command action. -->
<!ENTITY % default-command-action   '"allow"'>

<!-- Default rekey interval in seconds. -->
<!ENTITY % default-rekey-interval-seconds  '"3600"'>

<!-- Default rekey interval in bytes (1GB). -->
<!ENTITY % default-rekey-interval-bytes   '"1000000000"'>

<!-- Default login grace time seconds. -->
<!ENTITY % default-login-grace-time-seconds  '"600"'>

<!-- Default authentication action. -->
<!ENTITY % default-authentication-action  '"allow"'>

<!-- Password authentication failure delay. -->
<!ENTITY % default-auth-password-failure-delay  '"2"'>

<!-- Password authentication maximum tries. -->
<!ENTITY % default-auth-password-max-tries  '"3"'>

<!-- DNS match not required by default in hostbased. -->
<!ENTITY % default-auth-hostbased-require-dns-match '"no"'>

<!-- Keyboard-interactive authentication failure delay. -->
<!ENTITY % default-auth-kbdint-failure-delay  '"2"'>

<!-- Keyboard-interactive authentication maximum tries. -->
<!ENTITY % default-auth-kbdint-max-tries  '"3"'>

<!-- Keyboard-interactive RADIUS server default port. -->
<!ENTITY % default-radius-server-port   '"1812"'>

<!-- Keyboard-interactive RADIUS server default UDP recvfrom timeout. -->
<!ENTITY % default-radius-server-timeout  '"10"'>

<!-- GSS-API default ticket forwarding policy. -->
<!ENTITY % default-gssapi-ticket-forwarding-policy '"no"'>

<!-- Use-expired-crls in cert validation. -->
<!ENTITY % default-use-expired-crls   '"0"'>

<!-- Disable-crls in cert validation. -->
<!ENTITY % default-disable-crls    '"no"'>

<!-- Dod pki in cert validation. -->
<!ENTITY % default-dod-pki    '"no"'>

<!-- LDAP server default port. -->
<!ENTITY % default-ldap-server-port   '"389"'>
```

```
<!-- Default interval for CRL prefetching. -->
<!ENTITY % default-crl-prefetch-interval  '"3600"'>

<!-- Default crypto-lib mode ("fips" or "standard"). -->
<!ENTITY % default-crypto-lib-mode   '"standard"'>

<!-- Log event facility. -->
<!ENTITY % default-log-event-facility   '"normal"'>

<!-- Log event severity. -->
<!ENTITY % default-log-event-severity   '"notice"'>

<!-- Ignore AIX Rlogin setting. -->
<!ENTITY % default-aix-ignore-rlogin               '"no"'>

<!-- TCP keepalive option. -->
<!ENTITY % default-tcp-keepalive   '"no"'>

<!-- Default connection idle timeout in seconds.  The value zero -->
<!-- disables idle timeout. -->
<!ENTITY % default-idle-timeout    '"0"'>

<!-- Should the message of the day (MOTD) be printed on login. -->
<!ENTITY % default-print-motd    '"yes"'>

<!-- Should authentication file permissions be checked. -->
<!ENTITY % default-strict-modes    '"yes"'>

<!-- Default authentication file permission mask bits (octal). -->
<!ENTITY % default-file-mask-bits   '"022"'>


<!-- Policy elements. -->

<!-- The top-level element -->
<!ELEMENT secsh-server (params?,connections?,authentication-methods?
    ,services?)>

<!-- Parameter element. -->
<!ELEMENT params (crypto-lib?,settings?,hostkey*,listener*,logging?,
    limits?,cert-validation?)>

<!-- Crypto-lib. -->
<!ELEMENT crypto-lib EMPTY>
<!ATTLIST crypto-lib
   mode  (fips|standard) %default-crypto-lib-mode;>

<!-- Settings - a block for stuff that is too minor to have it's
     own element in the params block. -->
<!ELEMENT settings EMPTY>
```

```
<!ATTLIST settings
   proxy-scheme  CDATA  #IMPLIED
   xauth-path  CDATA  #IMPLIED
   ignore-aix-rlogin (yes|no) %default-aix-ignore-rlogin;
          user-config-dir CDATA  #IMPLIED>

<!-- Hostkey specification. -->
<!ELEMENT hostkey ((private,(public|x509-certificate)?)|externalkey)>

<!-- Private key specification. -->
<!ELEMENT private (#PCDATA)>
<!ATTLIST private
   file  CDATA #IMPLIED>

<!-- Public key. -->
<!ELEMENT public (#PCDATA)>
<!ATTLIST public
   file  CDATA #IMPLIED>

<!-- Certificate (host) -->
<!ELEMENT x509-certificate (#PCDATA)>
<!ATTLIST x509-certificate
   file  CDATA #IMPLIED>

<!-- External key. -->
<!ELEMENT externalkey EMPTY>
<!ATTLIST externalkey
   type  CDATA #REQUIRED
   init-info CDATA #IMPLIED>

<!-- CA certificate. -->
<!ELEMENT ca-certificate (#PCDATA)>
<!ATTLIST ca-certificate
   file   CDATA  #IMPLIED
   name   CDATA  #REQUIRED
   disable-crls  (yes|no) %default-disable-crls;
   use-expired-crls CDATA  %default-use-expired-crls;>

<!-- Certificate caching -->
<!ELEMENT cert-cache-file EMPTY>
<!ATTLIST cert-cache-file
   file   CDATA #REQUIRED>

<!-- CRL automatic updating -->
<!ELEMENT crl-auto-update EMPTY>
<!ATTLIST crl-auto-update
   update-before  CDATA #IMPLIED
   minimum-interval CDATA #IMPLIED>

<!-- CRL prefetch -->
<!ELEMENT crl-prefetch  EMPTY>
```

```
<!ATTLIST crl-prefetch
   interval  CDATA %default-crl-prefetch-interval;
   url   CDATA #REQUIRED>

<!-- LDAP server -->
<!ELEMENT ldap-server  EMPTY>
<!ATTLIST ldap-server
   address  CDATA #REQUIRED
   port   CDATA %default-ldap-server-port;>

<!-- OCSP responder -->
<!ELEMENT ocsp-responder EMPTY>
<!ATTLIST ocsp-responder
   validity-period CDATA #IMPLIED
   url   CDATA #REQUIRED>

<!-- Enable DOD PKI compliancy -->
<!ELEMENT dod-pki  EMPTY>
<!ATTLIST dod-pki
   enable (yes|no) %default-dod-pki;>

<!-- TCP listener. -->
<!ELEMENT listener EMPTY>
<!ATTLIST listener
   id  ID #REQUIRED
   port  CDATA "22"
   address CDATA #IMPLIED>

<!-- Logging. -->
<!ELEMENT logging (log-events*)>

<!-- Log-events -->
<!ELEMENT log-events (#PCDATA)>
<!ATTLIST log-events
   facility (normal|daemon|user|auth|local0|local1
    |local2|local3|local4|local5|local6|local7|discard)
   %default-log-event-facility;
   severity (informational|notice|warning|error|critical
    |security-success|security-failure)
   %default-log-event-severity;>

<!-- Cert-validation. -->
<!ELEMENT cert-validation (ldap-server*,ocsp-responder*,cert-cache-file?
      ,crl-auto-update?,crl-prefetch*,dod-pki?
      ,ca-certificate*)>

<!ATTLIST cert-validation
   http-proxy-url CDATA #IMPLIED
   socks-server-url CDATA #IMPLIED>

<!-- Limits. -->
```

```
<!ELEMENT limits  EMPTY>
<!ATTLIST limits
   max-connections CDATA #IMPLIED
   max-processes  CDATA #IMPLIED>

<!-- Connections. -->
<!ELEMENT connections (connection+)>

<!-- Connection. -->
<!ELEMENT connection (selector*,rekey?,cipher*,mac*)>
<!ATTLIST connection
   name  ID   #IMPLIED
   action (allow|deny)  %default-connection-action;
   tcp-keepalive (yes|no)  %default-tcp-keepalive;>

<!-- Rekey intervals. -->
<!ELEMENT rekey  EMPTY>
<!ATTLIST rekey
   seconds CDATA %default-rekey-interval-seconds;
   bytes  CDATA %default-rekey-interval-bytes;>

<!-- Cipher. -->
<!ELEMENT cipher EMPTY>
<!ATTLIST cipher
   name CDATA #REQUIRED>

<!-- MAC. -->
<!ELEMENT mac  EMPTY>
<!ATTLIST mac
   name CDATA #REQUIRED>

<!-- Selector element. -->
<!ELEMENT selector ((interface|certificate|host-certificate|ip
     |user|user-group|user-privileged|blackboard
     |publickey-passed|user-password-change-needed)*)>

<!-- Interface selector. At least one parameter must be given. If id is -->
<!-- set, the others MUST NOT be set. If id is not set, either or both -->
<!-- of address and port may be defined.    -->
<!ELEMENT interface EMPTY>
<!ATTLIST interface
   id  IDREF #IMPLIED
   address CDATA #IMPLIED
   port  CDATA #IMPLIED>

<!-- Publickey (plain) passed selector. -->
<!ELEMENT publickey-passed EMPTY>
<!ATTLIST publickey-passed
   length CDATA #IMPLIED>

<!-- Certificate selector. -->
```

```
<!ELEMENT certificate EMPTY>
<!ATTLIST certificate
   field    (ca-list|issuer-name|subject-name|serial-number
      |altname-email|altname-upn
      |altname-ip|altname-fqdn) #REQUIRED
   pattern   CDATA #IMPLIED
   pattern-case-sensitive CDATA #IMPLIED>

<!-- Host certificate selector. -->
<!ELEMENT host-certificate EMPTY>
<!ATTLIST host-certificate
   field    (ca-list|issuer-name|subject-name|serial-number
      |altname-email|altname-upn
      |altname-ip|altname-fqdn) #REQUIRED
   pattern   CDATA #IMPLIED
   pattern-case-sensitive CDATA #IMPLIED>

<!-- IP address selector. -->
<!-- The address will be one of the following:  -->
<!--   - an IP-range of the form x.x.x.x-y.y.y.y -->
<!--   - an IP-mask of the form x.x.x.x/y  -->
<!--   - a straight IP-address x.x.x.x   -->
<!--   - an FQDN pattern (form not checked,  -->
<!--  either it matches or not)   -->
<!-- Exactly one of address or fqdn must be set. -->
<!ELEMENT ip  EMPTY>
<!ATTLIST ip
   address CDATA #IMPLIED
   fqdn  CDATA #IMPLIED>

<!-- User name selector. -->
<!ELEMENT user   EMPTY>
<!ATTLIST user
   name   CDATA #IMPLIED
   name-case-sensitive CDATA #IMPLIED
   id   CDATA #IMPLIED>

<!-- User group selector. -->
<!ELEMENT user-group  EMPTY>
<!ATTLIST user-group
   name   CDATA #IMPLIED
   name-case-sensitive CDATA #IMPLIED
   id   CDATA #IMPLIED>

<!-- User privileged selector. -->
<!ELEMENT user-privileged EMPTY>
<!ATTLIST user-privileged
   value (yes|no) %default-user-privileged-value;>

<!-- Selector for the need of user password change. -->
<!ELEMENT user-password-change-needed EMPTY>
```

```
<!ATTLIST user-password-change-needed
   value (yes|no) %default-user-password-change-needed-value;>

<!-- Blackboard selector. -->
<!ELEMENT blackboard  EMPTY>
<!ATTLIST blackboard
   field    CDATA #REQUIRED
   pattern   CDATA #IMPLIED
   pattern-case-sensitive CDATA #IMPLIED>


<!-- Authentication methods element. -->
<!ELEMENT authentication-methods (banner-message?,auth-file-modes?
      ,authentication*)>
<!ATTLIST authentication-methods
   login-grace-time CDATA %default-login-grace-time-seconds;>

<!-- Banner-message element. -->
<!ELEMENT banner-message (#PCDATA)>
<!ATTLIST banner-message
   file  CDATA #IMPLIED>

<!-- Authentication file permission checks. -->
<!ELEMENT auth-file-modes EMPTY>
<!ATTLIST auth-file-modes
   strict  (yes|no) %default-strict-modes;
   mask-bits  CDATA  %default-file-mask-bits;>

<!-- Authentication element.  In an authentication element, different -->
<!-- authentication methods are in OR-relation.  User must pass one of -->
<!-- them. -->
<!ELEMENT authentication (selector*
     ,(auth-publickey|auth-hostbased|auth-password
       |auth-keyboard-interactive|auth-gssapi)*
     ,authentication*)>
<!ATTLIST authentication
   name  ID  #IMPLIED
   action (allow|deny) %default-authentication-action;
   set-group CDATA  #IMPLIED>

<!-- Publickey authentication. -->
<!ELEMENT auth-publickey EMPTY>

<!-- Hostbased authentication. -->
<!ELEMENT auth-hostbased EMPTY>
<!ATTLIST auth-hostbased
   require-dns-match (yes|no)
    %default-auth-hostbased-require-dns-match;>

<!-- Password authentication. -->
<!ELEMENT auth-password  EMPTY>
```

```
<!ATTLIST auth-password
   failure-delay  CDATA %default-auth-password-failure-delay;
   max-tries  CDATA %default-auth-password-max-tries;>

<!-- Keyboard interactive authentication. -->
<!ELEMENT auth-keyboard-interactive ((submethod-pam
       |submethod-password
       |submethod-securid
       |submethod-radius
       |submethod-generic)*)>

<!ATTLIST auth-keyboard-interactive
   failure-delay  CDATA %default-auth-kbdint-failure-delay;
   max-tries  CDATA %default-auth-kbdint-max-tries;>

<!-- Keyboard interactive sub-methods. -->

<!-- PAM. -->
<!ELEMENT submethod-pam  EMPTY>
<!ATTLIST submethod-pam
   dll-path  CDATA #IMPLIED>

<!-- Password. -->
<!ELEMENT submethod-password EMPTY>

<!-- SecurID. -->
<!ELEMENT submethod-securid EMPTY>
<!ATTLIST submethod-securid
   dll-path  CDATA #IMPLIED>

<!-- RADIUS. -->
<!ELEMENT submethod-radius (radius-server+)>

<!-- RADIUS server. -->
<!ELEMENT radius-server  (radius-shared-secret)>
<!ATTLIST radius-server
   address  CDATA #REQUIRED
   port   CDATA %default-radius-server-port;
   timeout  CDATA %default-radius-server-timeout;
   client-nas-identifier CDATA #IMPLIED>

<!-- Secret. "file" has precedence over #PCDATA. -->
<!ELEMENT radius-shared-secret (#PCDATA)>
<!ATTLIST radius-shared-secret
   file   CDATA #IMPLIED>

<!-- Generic submethod. -->
<!ELEMENT submethod-generic EMPTY>
<!ATTLIST submethod-generic
   name   CDATA #REQUIRED
   params  CDATA #IMPLIED>
```

```
<!-- GSS-API authentication. -->
<!ELEMENT auth-gssapi EMPTY>
<!ATTLIST auth-gssapi
   dll-path        CDATA #IMPLIED
   allow-ticket-forwarding     (yes|no)
  %default-gssapi-ticket-forwarding-policy;>

<!-- Services element. -->
<!ELEMENT services (group*,rule+)>

<!-- Group element. -->
<!ELEMENT group  (selector+)>
<!ATTLIST group
   name ID #REQUIRED>

<!-- Rule element. -->
<!ELEMENT rule  (environment*,terminal?,subsystem*,command*
    ,tunnel-agent?,tunnel-x11?,tunnel-local*
    ,tunnel-remote*)>

<!-- "group", if defined, will be used to match the rule. -->
<!ATTLIST rule
   group  CDATA  #IMPLIED
   idle-timeout CDATA  %default-idle-timeout;
   print-motd (yes|no) %default-print-motd;>

<!-- Environment. -->
<!-- The default allowed environment variables are:        -->
<!-- allowed-case-sensitive='"TERM,PATH,TZ,LANG,LC_*"'        -->
<!-- If neither allowed or allowed-case-sensitive is not set,  -->
<!-- the default is used.           -->
<!ELEMENT environment EMPTY>
<!ATTLIST environment
   allowed   CDATA #IMPLIED
   allowed-case-sensitive CDATA #IMPLIED>

<!-- Terminal. -->
<!ELEMENT terminal EMPTY>
<!ATTLIST terminal
   action (allow|deny)  %default-terminal-action;
   chroot CDATA   #IMPLIED>

<!-- Subsystem. -->
<!ELEMENT subsystem (attribute*)>
<!ATTLIST subsystem
   type  CDATA  #REQUIRED
   action (allow|deny) %default-subsystem-action;
   application CDATA  #IMPLIED
   chroot CDATA  #IMPLIED>
```

```
<!ELEMENT attribute EMPTY>
<!ATTLIST attribute
   name  CDATA #REQUIRED
   value  CDATA #IMPLIED>

<!-- Tunnel -->

<!ELEMENT tunnel-x11 EMPTY>
<!ATTLIST tunnel-x11
   action (allow|deny)  %default-tunnel-action;>

<!ELEMENT tunnel-agent EMPTY>
<!ATTLIST tunnel-agent
   action (allow|deny)  %default-tunnel-action;>

<!ELEMENT tunnel-local ((src|dst)*)>
<!ATTLIST tunnel-local
   action (allow|deny)  %default-tunnel-action;>

<!ELEMENT tunnel-remote ((src|listen)*)>
<!ATTLIST tunnel-remote
   action (allow|deny)  %default-tunnel-action;>

<!-- Tunnel selectors. These apply only to TCP local and remote -->
<!-- tunnels. -->
<!-- src and dst are for local-tcp -->
<!-- src and listen are for remote-tcp -->

<!-- Address or fqdn are not mandatory. If set, exactly one must be set -->
<!-- (not both).       -->

<!-- Source. -->
<!ELEMENT src  EMPTY>
<!ATTLIST src
   address CDATA #IMPLIED
   fqdn  CDATA #IMPLIED
   port  CDATA #IMPLIED>

<!-- Destination. -->
<!ELEMENT dst  EMPTY>
<!ATTLIST dst
   address CDATA #IMPLIED
   fqdn  CDATA #IMPLIED
   port  CDATA #IMPLIED>

<!-- Listener. -->
<!ELEMENT listen EMPTY>
<!ATTLIST listen
   address CDATA #IMPLIED
   port  CDATA #IMPLIED>
```

```
<!-- Command. -->
<!ELEMENT command   EMPTY>
<!ATTLIST command
   action   (allow|deny|forced)
      %default-command-action;
   application   CDATA #IMPLIED
   application-case-sensitive CDATA #IMPLIED
   chroot   CDATA #IMPLIED>
```

# Appendix C Man Pages and Help Files

On Unix, the following manual pages are included in the SSH Tectia Server distribution:

- **ssh-server-g3.8**: Secure Shell server – Generation 3
- **ssh-server-config.5**: Secure Shell server configuration file format
- **ssh-server-config-tool.8**: Secure Shell server configuration updater
- **ssh-keygen-g3.1**: authentication key pair generator
- **ssh-cmpclient-g3.1**: certificate enrollment client
- **ssh-certview-g3.1**: certificate viewer
- **ssh-ekview-g3.1**: external key viewer

On Windows, the **SSH Tectia Server** program group includes links to SSH Tectia user documentation in PDF format. The documents can be found in the `<INSTALLDIR>\SSH Tectia AUX\documents` directory.

# Appendix D Audit Messages

This appendix lists the audit messages generated by the server.

**100 Server_starting**
**Level:** notice
**Origin:** SSH Tectia Server

The server is starting.

Default log facility: daemon

**101 Server_start_failed**
**Level:** error
**Origin:** SSH Tectia Server

The server has encountered a fatal error condition and cannot proceed.

Default log facility: daemon

| Argument | Description |
|---|---|
| Success \| Error | Error code |
| Text | Description of the error |

**102 Server_running**
**Level:** notice
**Origin:** SSH Tectia Server

The server has started and is running normally.

Default log facility: daemon

**103 Server_stopping**
**Level:** notice
**Origin:** SSH Tectia Server

The server is shutting down.

Default log facility: daemon

**104 Server_exiting**
**Level:** notice
**Origin:** SSH Tectia Server

The server is exiting.

Default log facility: daemon

**105 Server_reconfig_started**
**Level:** informational
**Origin:** SSH Tectia Server

The server is starting the reconfiguration operation

Default log facility: daemon

| **Argument** | **Description** |
|---|---|
| File name | The name of the configuration file |

**106 Server_reconfig_finished**
**Level:** notice
**Origin:** SSH Tectia Server

The server reconfiguration has finished.

Default log facility: daemon

| **Argument** | **Description** |
|---|---|
| Success \| Error | Success or error code |
| Text | Textual description of the error |

**107 Server_listener_started**
**Level:** notice
**Origin:** SSH Tectia Server

The server successfully established a listener socket.

Default log facility: daemon

| **Argument** | **Description** |
|---|---|
| Listener | Address of the listener socket |
| Listener Port | Port of the listener socket |

**108 Server_listener_stopped**
**Level:** notice
**Origin:** SSH Tectia Server

The server closed a listener socket.

Default log facility: daemon

| **Argument** | **Description** |
|---|---|
| Listener | Address of the listener socket |
| Listener Port | Port of the listener socket |

**109 Server_listener_failed**
**Level:** warning
**Origin:** SSH Tectia Server

The server failed to open a listener socket.

Default log facility: daemon

| **Argument** | **Description** |
|---|---|
| Listener | Address of the listener socket |
| Listener Port | Port of the listener socket |

**110 Servant_exited**
**Level:** warning
**Origin:** SSH Tectia Server

A servant has exited, possibly because of an error.

Default log facility: daemon

| **Argument** | **Description** |
|---|---|
| Pid | Process ID of the servant |
| Success \| Error | Success or error code |
| Exit Value | Exit value of the servant process |

**111 Servant_error**
**Level:** warning
**Origin:** SSH Tectia Server

A servant reported an unrecoverable error

Default log facility: daemon

| **Argument** | **Description** |
|---|---|
| Pid | Process ID of the servant |
| Text | Textual error message |

**112 Server_error**
**Level:** warning
**Origin:** SSH Tectia Server

The crypto library FIPS status was changed in reconfiguration.

Default log facility: daemon

| Argument | Description |
|---|---|
| Text | Verbose description of the error |

**113 Server_warning**
**Level:** warning
**Origin:** SSH Tectia Server

The server encountered a non-fatal error.

Default log facility: daemon

| Argument | Description |
|---|---|
| Text | Description of the error |

**114 Servant_warning**
**Level:** warning
**Origin:** SSH Tectia Server

The server encountered a non-fatal error condition.

Default log facility: daemon

| Argument | Description |
|---|---|
| Text | Verbose warning message |

**115 Servant_info**
**Level:** informational
**Origin:** SSH Tectia Server

The server encountered a non-error condition of interest.

Default log facility: daemon

| Argument | Description |
|---|---|
| Text | Verbose informational message |

**116 Servant_client_verbose**
**Level:** notice
**Origin:** SSH Tectia Server

The Client sent a high priority debug message.

Default log facility: daemon

| Argument | Description |
|---|---|
| Text | Client debug message |

| Argument | Description |
|---|---|
| Session-Id | Session identifier |

**117 Servant_client_debug**
**Level:** informational
**Origin:** SSH Tectia Server

The Client sent a low priority debug message.

Default log facility: daemon

| Argument | Description |
|---|---|
| Text | Client debug message |
| Session-Id | Session identifier |

**400 Connect**
**Level:** security-success
**Origin:** SSH Tectia Server

The server initially accepted an incoming connection.

Default log facility: normal

| Argument | Description |
|---|---|
| Policy name | Symbolic name for the connection policy definition in the server configuration (optional) |
| Src | Remote hostname |
| Src IP | Remote IP address |
| Dst IFace | Local interface ID |
| Dst IP | Local IP address |
| Src Port | Remote port |
| Dst Port | Local port |
| Ver | Client's version string |

**401 Connection_denied**
**Level:** security-failure
**Origin:** SSH Tectia Server

Connection was denied because maximum number of connections was reached.

Default log facility: normal

| Argument | Description |
|---|---|
| Success \| Error | Reason for the connection being denied |
| Src IP | Remote IP address |
| Dst IFace | Local interface ID |
| Dst IP | Local IP address |

| Argument | Description |
|---|---|
| Src Port | Remote port |
| Dst Port | Local port. |

**402 Disconnect**
**Level:** informational
**Origin:** SSH Tectia Server

The connection has closed.

Default log facility: normal

| Argument | Description |
|---|---|
| Reason | Reason for disconnect |
| Src | Remote hostname |
| Src IP | Remote IP address |
| Dst IFace | Local interface ID |
| Dst IP | Local IP address |
| Src Port | Remote port |
| Dst Port | Local port |
| Text | Textual description of the disconnect reason |

**410 Login_success**
**Level:** security-success
**Origin:** SSH Tectia Server

The user has logged in after successful authentication, account validity and other checks.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | The user's login name |
| Src | Remote hostname |
| Src IP | Remote IP address |
| Dst IFace | Local interface ID |
| Dst IP | Local IP address |
| Src Port | Remote port |
| Dst Port | Local port |
| Ver | Client's version string |
| Session-Id | Session identifier |

**411 Login_failure**
**Level:** security-failure
**Origin:** SSH Tectia Server

An unsuccessful login attempt was made to the server.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | The user's login name |
| Reason | Reason for disconnect |
| Src | Remote hostname |
| Src IP | Remote IP address |
| Dst IFace | Local interface ID |
| Dst IP | Local IP address |
| Src Port | Remote port |
| Dst Port | Local port |
| Text | Textual description of the disconnect reason |
| Session-Id | Session identifier |

**412 Logout**
**Level:** informational
**Origin:** SSH Tectia Server

The user has logged out.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | The user's login name |
| Reason | Reason for disconnect |
| Src | Remote hostname |
| Src IP | Remote IP address |
| Dst IFace | Local interface ID |
| Dst IP | Local IP address |
| Src Port | Remote port |
| Dst Port | Local port |
| Text | Textual description of the disconnect reason |
| Session-Id | Session identifier |

**420 Session_channel_open**
**Level:** informational
**Origin:** SSH Tectia Server

A session channel opening has completed successfully or unsuccessfully.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Success \| Error | Success or error description |
| Command | The requested command |

| Argument | Description |
|---|---|
| Sub ID | Channel number |
| Session-Id | Session identifier |

**421 Session_channel_close**
**Level:** informational
**Origin:** SSH Tectia Server

A session channel has closed.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Sub ID | Channel number |
| Session-Id | Session identifier |

**422 Forwarding_channel_open**
**Level:** informational
**Origin:** SSH Tectia Server

A TCP forwarding channel request to the client has completed successfully or unsuccessfully.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Success \| Error | Success or error description |
| Tunnel type | "Server to client" |
| Listener IP | Receiving listener address |
| Listener Port | Receiving listener port |
| Sub ID | Channel number |
| Session-Id | Session identifier |
| Text | Error message |

**423 Forwarding_channel_close**
**Level:** informational
**Origin:** SSH Tectia Server

A TCP forwarding channel has closed.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Sub ID | Channel number |
| Session-Id | Session identifier |

**424 Auth_channel_open**
**Level:** informational
**Origin:** SSH Tectia Server

An authorization agent forwarding channel opening has completed successfully or unsuccessfully.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Sub ID | Channel number |
| File name | Path to the listener |
| Success \| Error | Success or error description |
| Session-Id | Session identifier |

**425 Auth_channel_close**
**Level:** informational
**Origin:** SSH Tectia Server

An authorization agent forwarding channel has closed.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Sub ID | Channel number |
| File name | Path to the listener |
| Session-Id | Session identifier |

**426 Forwarding_listener_open**
**Level:** informational
**Origin:** SSH Tectia Server

A TCP forwarding listener opening has completed successfully or unsuccessfully.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Success \| Error | Success or error description |
| Listener IP | Listener's IP address |
| Listener Port | Listener's port |
| Session-Id | Session identifier |
| Text | Error description |

**427 Forwarding_listener_close**
**Level:** informational

**Origin:** SSH Tectia Server

Trying to destroy a non-existing TCP listener.

Default log facility: normal

| **Argument** | **Description** |
|---|---|
| Username | User's login name |
| Listener IP | Listener's IP address |
| Listener Port | Listener's port |
| Session-Id | Session identifier |
| Success \| Error | Error description |
| Text | Error message. |

**428 Auth_listener_open**
**Level:** informational
**Origin:** SSH Tectia Server

An authorization agent forwarding listener opening has completed successfully or unsuccessfully.

Default log facility: normal

| **Argument** | **Description** |
|---|---|
| Username | User's login name |
| File name | Path to the listener |
| Success \| Error | Success or error description |
| Session-Id | Session identifier |

**429 Auth_listener_close**
**Level:** informational
**Origin:** SSH Tectia Server

An authorization agent forwarding listener has closed.

Default log facility: normal

| **Argument** | **Description** |
|---|---|
| Username | User's login name |
| File name | Path to the listener |
| Session-Id | Session identifier |

**430 Channel_open_failure**
**Level:** informational
**Origin:** SSH Tectia Server

Opening a channel has failed, because the other end has returned a failure.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Sub ID | Channel number |
| Failure code | Error code from the other end |
| Text | Textual description of the error from the other end. |
| Session-Id | Session identifier |

**431 X11_listener_open**
**Level:** informational
**Origin:** SSH Tectia Server

An X11 forwarding listener opening has completed successfully or unsuccessfully.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Display | Display name |
| Success \| Error | Success or error description |
| Text | Textual description of the error (optional) |
| Session-Id | Session identifier |

**432 X11_listener_close**
**Level:** informational
**Origin:** SSH Tectia Server

An X11 forwarding listener has closed.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Display | Display name |
| Session-Id | Session identifier |

**433 X11_channel_open**
**Level:** informational
**Origin:** SSH Tectia Server

An X11 forwarding channel opening has completed successfully or unsuccessfully.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |

| Argument | Description |
|---|---|
| Sub ID | Channel number |
| Display | Display name |
| Success \| Error | Success or error description |
| Text | Extra textual information (optional) |
| Session-Id | Session identifier |

**434 X11_channel_close**
**Level:** informational
**Origin:** SSH Tectia Server

An X11 forwarding channel has closed.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Sub ID | Channel number |
| Session-Id | Session identifier |

**460 Session_env_request_failure**
**Level:** informational
**Origin:** SSH Tectia Server

A session channel request to set an environment variable has failed.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Success \| Error | Success or error description |
| Text | Description of the error |
| Sub ID | Channel number |
| Session-Id | Session identifier |

**461 Session_env_file_failure**
**Level:** informational
**Origin:** SSH Tectia Server

When reading the user's environment file, an illegal environment variable was encountered.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| Success \| Error | Success or error description |
| File name | Environment file name |

| Argument | Description |
|---|---|
| Text | Description of the error |
| Session-Id | Session identifier |

**700 Auth_method_success**
**Level:** informational
**Origin:** SSH Tectia Server

An authentication method was successfully completed.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Auth method | Authentication method's name |
| Session-Id | Session identifier |

**701 Auth_method_failure**
**Level:** informational
**Origin:** SSH Tectia Server

An authentication method failed.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Auth method | Authentication method's name |
| Session-Id | Session identifier |

**702 Auth_methods_completed**
**Level:** informational
**Origin:** SSH Tectia Server

The cumulative list of completed authentication methods

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Auth methods | A list of completed authentication methods |
| Session-Id | Session identifier |

**703 Auth_methods_available**
**Level:** informational
**Origin:** SSH Tectia Server

The list of authentication methods still available for the user

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Auth methods | A list of available authentication methods |
| Session-Id | Session identifier |

**705 Auth_error**
**Level:** warning
**Origin:** SSH Tectia Server

An error occurred during authentication.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Text | Textual description of the error |
| Session-Id | Session identifier |

**706 Auth_warning**
**Level:** informational
**Origin:** SSH Tectia Server

Something abnormal happened during authentication, but authentication can still continue.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Text | Textual description of the error |
| Session-Id | Session identifier |

**707 Publickey_auth_success**
**Level:** informational
**Origin:** SSH Tectia Server

The user successfully completed the publickey authentication method.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | 'publickey' |
| Text | Acceptance description |

| Argument | Description |
|---|---|
| Session-Id | Session identifier |

**708 Publickey_auth_error**
**Level:** warning
**Origin:** SSH Tectia Server

Error in the publickey authentication method. This means that the current user authentication attempt with the publickey method has failed.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | 'publickey' |
| Text | Error description |
| Session-Id | Session identifier |

**709 Publickey_auth_warning**
**Level:** informational
**Origin:** SSH Tectia Server

Warning during the publickey authentication method. This means that something abnormal happened during the publickey method, but the method may still complete successfully.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | 'publickey' |
| Text | Warning description |
| Session-Id | Session identifier |

**711 Hostbased_auth_error**
**Level:** warning
**Origin:** SSH Tectia Server

Error in the hostbased authentication method. This means that the current user authentication attempt with the hostbased method has failed.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | Authentication method name |
| Text | Error description |
| Session-Id | Session identifier |

**712 Hostbased_auth_warning**
**Level:** informational
**Origin:** SSH Tectia Server

Warning during the hostbased authentication method. This means that something abnormal happened during the hostbased authentication method, but the method may still complete successfully.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | Authentication method name |
| Text | Warning description |
| Session-Id | Session identifier |

**714 Password_auth_error**
**Level:** warning
**Origin:** SSH Tectia Server

Error in the password authentication method. This means that the current user authentication attempt with the password method has failed.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | 'password' |
| Text | Error description |
| Session-Id | Session identifier |

**715 Password_auth_warning**
**Level:** informational
**Origin:** SSH Tectia Server

Warning during the password authentication method. This means that something abnormal happened in the password method, but the method may still complete successfully.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | 'password' |
| Text | Warning description |
| Session-Id | Session identifier |

**716 Keyboard_interactive_pam_auth_success**
**Level:** informational

**Origin:** SSH Tectia Server

The user successfully completed the Keyboard-interactive PAM authentication method.

Default log facility: auth

| **Argument** | **Description** |
|---|---|
| Username | User's login name |
| Algorithm | 'Keyboard-interactive PAM' |
| Text | Acceptance description |
| Session-Id | Session identifier |

**717 Keyboard_interactive_pam_auth_error**
**Level:** warning
**Origin:** SSH Tectia Server

Error in the Keyboard-interactive PAM authentication method. This means that the current user authentication attempt with the Keyboard-interactive PAM method has failed.

Default log facility: auth

| **Argument** | **Description** |
|---|---|
| Username | User's login name |
| Algorithm | Authentication method name |
| Text | Error description |
| Session-Id | Session identifier |

**718 Keyboard_interactive_pam_auth_warning**
**Level:** informational
**Origin:** SSH Tectia Server

Warning during the Keyboard-interactive PAM authentication method. This means that something abnormal happened during the Keyboard-interactive PAM authentication method, but the method may still complete successfully.

Default log facility: auth

| **Argument** | **Description** |
|---|---|
| Username | User's login name |
| Algorithm | Authentication method name |
| Text | Warning description |
| Session-Id | Session identifier |

**719 Keyboard_interactive_radius_auth_success**
**Level:** informational
**Origin:** SSH Tectia Server

The user successfully completed the Keyboard-interactive Radius authentication method.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | 'Keyboard-interactive Radius' |
| Text | Acceptance description |
| Session-Id | Session identifier |

**720 Keyboard_interactive_radius_auth_error**
**Level:** warning
**Origin:** SSH Tectia Server

Error in the Keyboard-interactive Radius authentication method. This means that the current user authentication attempt with the Keyboard-interactive Radius method has failed.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | Authentication method name |
| Text | Error description |
| Session-Id | Session identifier |

**721 Keyboard_interactive_radius_auth_warning**
**Level:** informational
**Origin:** SSH Tectia Server

Warning during the Keyboard-interactive Radius authentication method. This means that something abnormal happened during the Keyboard-interactive Radius authentication method, but the method may still complete successfully.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | Authentication method name |
| Text | Warning description |
| Session-Id | Session identifier |

**722 Keyboard_interactive_password_auth_success**
**Level:** informational
**Origin:** SSH Tectia Server

The user successfully completed the Keyboard-interactive Password authentication method.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | 'Keyboard-interactive Password' |
| Text | Acceptance description |
| Session-Id | Session identifier |

**723 Keyboard_interactive_password_auth_error**
**Level:** warning
**Origin:** SSH Tectia Server

Error in the Keyboard-interactive password authentication method. This means that the current user authentication attempt with the Keyboard-interactive password method has failed.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | Authentication method name |
| Text | Error description |
| Session-Id | Session identifier |

**725 Keyboard_interactive_securid_auth_success**
**Level:** informational
**Origin:** SSH Tectia Server

The user successfully completed the Keyboard-interactive SecurID authentication method.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | 'Keyboard-interactive SecurID' |
| Text | Acceptance description |
| Session-Id | Session identifier |

**726 Keyboard_interactive_securid_auth_error**
**Level:** warning
**Origin:** SSH Tectia Server

Error in the Keyboard-interactive SecurID authentication method. This means that the current user authentication attempt with the Keyboard-interactive SecurID method has failed.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | Authentication method name |
| Text | Error description |
| Session-Id | Session identifier |

**727 Keyboard_interactive_securid_auth_warning**
**Level:** informational
**Origin:** SSH Tectia Server

Warning during the Keyboard-interactive SecurID authentication method. This means that something abnormal happened during the Keyboard-interactive SecurID authentication method, but the method may still complete successfully.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | Authentication method name |
| Text | Warning description |
| Session-Id | Session identifier |

**728 GSSAPI_auth_success**
**Level:** informational
**Origin:** SSH Tectia Server

The user successfully completed the GSSAPI authentication method.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | 'GSSAPI' |
| Text | Acceptance description |
| Session-Id | Session identifier |

**729 GSSAPI_auth_error**
**Level:** warning
**Origin:** SSH Tectia Server

Error in the GSSAPI authentication method. This means that the current user authentication attempt with the GSSAPI method has failed.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |

| Argument | Description |
|---|---|
| Algorithm | 'publickey' |
| Text | Error description |
| Session-Id | Session identifier |

**730 GSSAPI_auth_warning**
**Level:** informational
**Origin:** SSH Tectia Server

Warning during the GSSAPI authentication method. This means that something abnormal happened during the GSSAPI method, but the method may still complete successfully.

Default log facility: auth

| Argument | Description |
|---|---|
| Username | User's login name |
| Algorithm | 'publickey' |
| Text | Warning description |
| Session-Id | Session identifier |

**800 Rule_engine_failure**
**Level:** error
**Origin:** SSH Tectia Server

An internal error occurred in rule engine.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | The user's login name, if available. |
| Policy name | Current authentication- or rule-element's symbolic name, if available. |
| Text | Textual representation of the error. |
| Session-Id | Session id, if available. |

**801 Authentication_block_selected**
**Level:** informational
**Origin:** SSH Tectia Server

The server selected an authentication block in the configuration.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | The user's login name |
| Policy name | A symbolic name for the authentication block |
| Session-Id | Session identifier |

**802 Authentication_block_allow**
**Level:** informational
**Origin:** SSH Tectia Server

The authentication chain terminated into an authentication block with an allow action.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | The user's login name |
| Policy name | A symbolic name for the authentication block |
| Session-Id | Session identifier |

**803 Authentication_block_deny**
**Level:** informational
**Origin:** SSH Tectia Server

The authentication chain terminated into an authentication block with a deny action.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | The user's login name |
| Policy name | A symbolic name for the authentication block |
| Session-Id | Session identifier |

**804 Group_selected**
**Level:** informational
**Origin:** SSH Tectia Server

The connection matched a group in the configuration.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | The user's login name |
| Policy name | A symbolic name for the group |
| Session-Id | Session identifier |

**805 Rule_selected**
**Level:** informational
**Origin:** SSH Tectia Server

The server selected a rule in the configuration.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | The user's login name |
| Policy name | A symbolic name for the rule |
| Session-Id | Session identifier |

**1000 KEX_failure**
**Level:** warning
**Origin:** SSH Tectia Server, Connection Broker

The key exchange failed.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name (not present for first KEX) |
| Algorithm | KEX algorithm name (not present if failure happens before choosing the algorithm) |
| Text | Error description |
| Session-Id | Session identifier (not present for first KEX) |

**1001 Algorithm_negotiation_failure**
**Level:** warning
**Origin:** SSH Tectia Server, Connection Broker

Algorithm negotiation failed - there was no common algorithm in the client's and server's lists.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name (not present for first KEX) |
| Algorithm | Algorithm type |
| Client algorithms | Client's algorithm list |
| Server algorithms | Server's algorithm list |
| Session-Id | Session identifier (not present for first KEX) |

**1002 Algorithm_negotiation_success**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

Algorithm negotiation succeeded.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name (not present for first KEX) |
| Text | Negotiated algorithms |
| Session-Id | Session identifier (not present for first KEX) |

**1100 Certificate_validation_failure**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

A received certificate failed to validate correctly under any of the configured CAs.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name (not present for first KEX) |
| Text | Resulting search states for all configured CAs. |
| Session-Id | Session identifier (not present for first KEX) |

**1101 Certificate_validation_success**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

A received certificate validated correctly under one or more configured CAs.

Default log facility: normal

| Argument | Description |
|---|---|
| Username | User's login name |
| CA List | A list of CAs under which the user's certificate validated correctly. |
| Session-Id | Session identifier |

**1110 CM_find_started**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

A low-level search was started in the certificate validation subsystem.

Default log facility: normal

| Argument | Description |
|---|---|
| Ctx | Search context |
| Search constraints | Search constraints. |

**1111 CM_find_finished**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

A low-level find operation has finished in the certificate validation subsystem.

Default log facility: normal

| Argument | Description |
|---|---|
| Ctx | Context pointer that identifies the search |

**1112 CM_cert_not_in_search_interval**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

The certificate is not valid during the required time period.

Default log facility: normal

| Argument | Description |
|---|---|
| SubjectName | Subject name of the certificate |
| Text | Error description |
| Ctx | Search context |

**1113 CM_certificate_revoked**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

A certificate was found to be revoked.

Default log facility: normal

| Argument | Description |
|---|---|
| SubjectName | Subject name of the certificate |
| Ctx | The context pointer of the search |

**1114 CM_cert_search_constraint_mismatch**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

The certificate did not satisfy the constraints set for the search.

Default log facility: normal

| Argument | Description |
|---|---|
| SubjectName | Subject name of the certificate |
| Text | Description of the mismatch |
| Ctx | Search context |

**1115 CM_ldap_search_started**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

An LDAP search for a CRL or a sub-CA is being started.

Default log facility: normal

| Argument | Description |
|---|---|
| Text | Search details |

**1116 CM_ldap_search_success**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

An LDAP search for a CRL or a sub-CA completed successfully.

Default log facility: normal

| Argument | Description |
|---|---|
| Text | Search details |

**1117 CM_ldap_search_failure**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

The attempt to contact an LDAP server was unsuccessful.

Default log facility: normal

| Argument | Description |
|---|---|
| Text | Error details |

**1118 CM_http_search_started**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

The certificate validation subsystem is initiating a search for a CRL or a sub-CA through the HTTP protocol.

Default log facility: normal

| Argument | Description |
|---|---|
| Text | Search target |

**1119 CM_http_search_success**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

An HTTP request for a CRL or a sub-CA completed successfully.

Default log facility: normal

| Argument | Description |
|---|---|
| Text | Status message detailing what was being retrieved |

**1120 CM_http_search_failure**

**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

An HTTP request for a CRL or a sub-CA failed.

Default log facility: normal

| **Argument** | **Description** |
|---|---|
| Text | Error details |

**1121 CM_crl_added**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

A new CRL was successfully added to the certificate validation subsystem.

Default log facility: normal

| **Argument** | **Description** |
|---|---|
| Text | CRL's issuer and validity period |

**1200 Key_store_create**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

Key store created.

Default log facility: normal

**1201 Key_store_create_failed**
**Level:** warning
**Origin:** SSH Tectia Server, Connection Broker

Key store creation failed.

Default log facility: normal

**1202 Key_store_destroy**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

Key store destroyed.

Default log facility: normal

**1204 Key_store_add_provider**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

Added a provider to the key store.

Default log facility: normal

| Argument | Description |
|---|---|
| Type | Provider type |
| Init info | Initialization info |

**1205 Key_store_add_provider_failed**
**Level:** warning
**Origin:** SSH Tectia Server, Connection Broker

Adding a provider to the key store failed.

Default log facility: normal

| Argument | Description |
|---|---|
| Type | Provider type |
| Init info | Initialization info |
| EK error | Error message |

**1208 Key_store_decrypt**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

A key was used successfully for decryption.

Default log facility: normal

| Argument | Description |
|---|---|
| Key path | Key path |
| Fwd path | Fwd path |

**1209 Key_store_decrypt_failed**
**Level:** warning
**Origin:** SSH Tectia Server, Connection Broker

A key was used unsuccessfully for decryption.

Default log facility: normal

| Argument | Description |
|---|---|
| Key path | Key path |
| Fwd path | Fwd path |
| Crypto error | Error string |

**1210 Key_store_sign**
**Level:** informational

**Origin:** SSH Tectia Server, Connection Broker

A key was used successfully for signing.

Default log facility: normal

| **Argument** | **Description** |
|---|---|
| Key path | Key path |
| Fwd path | Fwd path |

**1211 Key_store_sign_failed**
**Level:** warning
**Origin:** SSH Tectia Server, Connection Broker

A key was used unsuccessfully for signing.

Default log facility: normal

| **Argument** | **Description** |
|---|---|
| Key path | Key path |
| Fwd path | Fwd path |
| Crypto error | Error string |

**1212 Key_store_sign_digest**
**Level:** informational
**Origin:** SSH Tectia Server, Connection Broker

A key was used successfully for signing a digest.

Default log facility: normal

| **Argument** | **Description** |
|---|---|
| Key path | Key path |
| Fwd path | Fwd path |

**1213 Key_store_sign_digest_failed**
**Level:** warning
**Origin:** SSH Tectia Server, Connection Broker

A key was used unsuccessfully for signing a digest.

Default log facility: normal

| **Argument** | **Description** |
|---|---|
| Key path | Key path |
| Fwd path | Fwd path |
| Crypto error | Error string |

# Index

## A

account
  local, 85
ACE/Agent, 95
ACE/Server, 95
address
  listen, 33, 59
administrators, 99
agent forwarding, 118
AIX
  installation, 16
  uninstallation, 20
allowed hosts, 100
allowing commands, 49, 77
allowing subsystems, 48, 76
allowing terminal access, 48, 78
allowing tunneling, 49–50, 77–78
application tunneling, 111
audit message reference, 151
auditing, 101
authentication, 69, 79
  certificate, 82, 86–87
  GSSAPI, 96
  host-based, 89
  host-based with certificates, 92
  Kerberos, 96
  keyboard-interactive, 93–96
  PAM, 93–94
  password, 84, 93
  public-key, 79, 85
  RADIUS, 93, 96
  SecurID, 93, 95
authentication chains, 40, 70
authentication methods, 39, 71, 79
authority info access, 82, 87
automated file transfer, 109

## B

banner message, 40, 55, 102
base-64, 126
basic configuration, 27

## C

CA certificate, 36, 64, 87
certificate
  enrolling, 83
  revoked, 82
certificate authentication
  server, 33, 56, 82
  user, 35, 61, 86–87
certificate cache file, 36, 62
certificate revocation list (CRL), 82, 87
certificate validation, 35, 61
certificate viewer, 125
certificates in host-based authentication, 92
certification
  FIPS 140-2, 31, 55
certification authority (CA), 35, 61, 82
changing host key, 81
channel, 111
chroot, 86
ciphers, 38, 69
CMP client, 128
command-line tools, 119
commands, 49, 77
Compatibility Notes, 26
configuration file
  server, 23, 27
  syntax, 137
connection rules, 37, 68
connections
  maximum number, 34, 54
CRL
  disabling, 36, 64
CRL auto update, 36, 63
CRL distribution point, 82, 87
CRL prefetch, 36, 63
CryptiCore, 106
cryptographic library, 31, 55
customer support, 10

D

default port, 33, 59
default settings, 53
denying commands, 49, 77
denying connection attempts, 100
denying file transfers, 114
denying subsystems, 48, 76
denying terminal access, 48, 78, 107, 113
denying tunneling, 49–50, 77–78, 107
Diffie-Hellman key exchange, 82
directory
  profile, 86
  root, 86
  virtual, 108
disabling CRL, 36, 64
disclaimer before login, 102
disk space requirement, 13
Document Type Definition (DTD), 137
documentation, 8
documentation conventions, 10
DOD PKI, 36, 63
domain controller, 85
domain user account, 85
DSA key pair, 80

E

editing selectors, 64
enrolling host certificate, 83
environment variables, 48, 76
event log, 34, 53, 60, 101, 151
examples of using SSH Tectia Server, 26
expired CRL, 36, 64
external host key, 33, 58, 84
external key viewer, 135

F

Federal Information Processing Standard (FIPS), 31, 55
file locations
  installed files, 23
file transfer, 105
  automated, 109
fingerprint, 81, 124
FIPS 140-2 certification, 31, 55
FIPS mode, 38–39, 55
firewall, 87
forced commands, 49, 77, 100
forwarding, 111
  agent, 118
  local, 114
  remote, 116
  X11, 117
fully qualified domain name (FQDN), 66, 83, 91

G

generating host key, 57, 81
Generic Security Service API (GSSAPI), 96
getting started with SSH Tectia Server, 23
getting support, 10
group, 41, 46, 71, 74, 85
GSSAPI authentication, 45, 72, 96

H

help files, 149
Hexl, 126
host certificate, 33, 57
  enrolling, 83
host key, 23, 32, 56
  changing, 81
  external, 33, 58, 84
  multiple, 80
  private, 32, 57
  public, 33, 57
host key generation, 57, 81
host-based authentication, 43, 72, 89
host-based authentication with certificates, 92
HP-UX
  installation, 17
  uninstallation, 21
HTTP proxy URL, 35, 62
HTTP repository, 82, 87

I

IBM AIX, 16

IBM Linux, 18
identity, 56
ignoring AIX rlogin restriction, 32
incoming tunnel, 111, 116
installation
    planning, 13
    upgrading, 15
installation packages, 14
installed files, 23
installing on AIX, 16
installing on HP-UX, 17
installing on Linux, 18
installing on Solaris, 18
installing on Windows, 19

## K

keepalive, 37, 69
Kerberos authentication, 96
key
    host, 32, 56
key exchange, 82
key fingerprint, 81, 124
key generation, 81
keyboard-interactive authentication, 43, 72, 93–96

## L

LDAP servers, 35, 63, 87
legal disclaimer, 102
library
    cryptographic, 31, 55
library certification
    FIPS 140-2, 31, 55
license file, 14, 23
licensing, 14
Lightweight Directory Access Protocol (LDAP), 82, 87
Linux
    installation, 18
    uninstallation, 21
Linux on POWER, 18
Linux on zSeries, 18
listen address, 33, 59, 100
listener, 33, 59
local port forwarding, 114
local tunnel, 114
local user account, 85
location
    installed files, 23
log message reference, 151
log on locally rights, 85
logging, 34, 53, 60, 101, 151
    customizing, 102
login grace time, 39, 55

## M

MACs, 39, 69
man pages, 149
man-in-the-middle attack, 82
maximum number of connections, 34, 54
maximum number of processes, 34, 54
message before login, 102
Microsoft Windows, 19
MSI package, 19
multiple host keys, 80

## N

Network Address Translation (NAT), 91
network interface binding, 100
network interface card, 59
network settings, 58

## O

OCSP responders, 36, 63, 87
Online Certificate Status Protocol (OCSP), 82, 87
outgoing tunnel, 111, 114

## P

PAM authentication, 44, 93–94
password authentication, 43–44, 71–72, 84–85, 93
PEM encoding, 126
Personal Information Exchange (PFX), 57
PKCS #7 package, 87
planning the installation, 13
Pluggable Authentication Module (PAM), 94

port forwarding, 111
  local, 114
  remote, 116
  restricting, 111
port number
  default, 33, 59
private key
  host, 32, 57, 80, 83
privileged users, 99
processes
  maximum number, 34, 54
profile directory, 86
proxy scheme, 31, 55
proxy server, 87
public key
  host, 33, 57
  user, 85
public-key authentication
  server, 32, 56, 79
  user, 43, 85

## R

RADIUS authentication, 44, 72, 93, 96
Red Hat Linux, 18
rekeying interval, 38, 69
related documents, 8
remote administration, 99
remote port forwarding, 116
remote tunnel, 116
removing from AIX, 20
removing from HP-UX, 21
removing from Linux, 21
removing from Solaris, 22
removing from Windows, 22
restoring default settings, 53
restricting services, 47, 75, 107, 112
restricting tunneling, 49–50, 77–78, 111–112
revoked certificate, 82
rights
  log on locally, 85
root directory, 86
RPM packages, 18
RSA ACE/Agent, 95
RSA ACE/Server, 95
RSA key pair, 80
RSA SecurID, 95

## S

secure application connectivity, 111
secure file transfer, 105
Secure File Transfer Protocol (SFTP), 86
Secure Shell server
  starting, 24
  stopping, 24
secure system administration, 99
SecurID authentication, 44, 72, 93, 95
selector
  blackboard, 42, 67
  certificate, 41, 66
  host certificate, 42, 66
  interface, 38, 42, 65
  IP, 38, 42, 66
  public key passed, 43, 67
  user, 42, 67
  user group, 42, 67
  user password change needed, 43
  user privileged, 42, 67
selector handling rules, 29
selectors, 28, 64
server
  starting, 24
  stopping, 24
server authentication methods, 32, 56, 79
server authentication with certificates, 82
server authentication with external key, 84
server authentication with public key, 79
server banner message, 102
server certificate, 33, 57, 82
server configuration file, 23, 27
server host key, 23
server settings, 52
server status, 53
server versions, 7
services

restricting, 47, 75, 107, 112
setting users to a group, 41, 46, 71, 74
settings
default, 53
SFT subsystem, 107
shared user account, 112
shell access, 99
socket, 59
SOCKS server URL, 35, 62
Solaris
installation, 18
uninstallation, 22
SSH Tectia Client, 9
SSH Tectia Client (F), 9
SSH Tectia Connector, 9
SSH Tectia Server, 9
starting, 24
stopping, 24
SSH Tectia Server (A), 9
SSH Tectia Server (F), 10
SSH Tectia Server (M), 10
SSH Tectia Server (T), 10
SSH Tectia Server Configuration tool, 52
SSH Tectia Server versions, 7
ssh-certview-g3(.exe), 125
ssh-cmpclient-g3(.exe), 128
ssh-ekview-g3(.exe), 135
ssh-keygen-g3(.exe), 81, 122
ssh-server-config-tool(.exe), 121
ssh-server-config.xml, 23, 27
ssh-server-g3(.exe), 119
starting the server, 24
status, 53
stopping the server, 24
subsystems, 48, 76
Sun Solaris, 18
supported platforms, 13
SUSE LINUX, 18
system configuration, 27
system log, 34, 53, 60, 101
system requirements, 13

**T**
technical support, 10
terminal access, 48, 78, 99
terminology, 8
ticket forwarding, 45, 72
trusted CA, 87
tunnel
agent, 49, 118
incoming, 116
local, 49, 114
outgoing, 114
remote, 50, 116
X11, 49, 117
tunneling, 111
access control, 112
restricting, 111

**U**
uninstalling from AIX, 20
uninstalling from HP-UX, 21
uninstalling from Linux, 21
uninstalling from Solaris, 22
uninstalling from Windows, 22
upgrading to 5.0
from 4.x, 15
from 5.0, 15
use cases, 23
user account
domain, 85
local, 85
shared, 112
user authentication based on host, 89, 92
user authentication methods, 39, 69, 79
user authentication with certificates, 86–87
user authentication with GSSAPI, 96
user authentication with keyboard-interactive, 93
user authentication with password, 84
user authentication with public key, 85
user configuration directory, 32, 56
user group, 85
User Manager, 85
user profile directory, 86

## V

viewing event log, 53
virtual directories, 108

## W

well-known port, 111
Windows
  installation, 19
  uninstallation, 22
Windows Event Log, 53, 60, 101
Windows password, 84–85
Windows user group, 85
Windows User Manager, 85

## X

X.509 certificate, 83, 87
X11 forwarding, 117
XAuth path, 32
XML attribute
  allow-ticket-forwarding, 45
  client-nas-identifier, 44
  disable-crls, 36
  dll-path, 44–45
  failure-delay, 43
  idle-timeout, 47
  ignore-aix-rlogin, 31
  login-grace-time, 39
  max-connections, 34
  max-processes, 34
  max-tries, 43
  print-motd, 47
  proxy-scheme, 31
  require-dns-match, 43
  set-group, 41
  tcp-keepalive, 37
  use-expired-crls, 36
  user-config-dir, 31
  xauth-path, 31
XML DTD, 27, 137
XML element, 27
  attribute, 48
  auth-file-modes, 40, 106
  auth-gssapi, 45
  auth-hostbased, 43
  auth-keyboard-interactive, 43
  auth-password, 43
  auth-publickey, 43
  authentication, 40, 45
  authentication-methods, 39
  banner-message, 40
  ca-certificate, 36
  cert-cache-file, 36
  cert-validation, 35
  cipher, 38
  command, 49
  connection, 37
  connections, 37
  crl-auto-update, 36
  crl-prefetch, 36
  crypto-lib, 31
  dod-pki, 36
  environment, 48
  externalkey, 33
  group, 46
  hostkey, 32
  http-proxy-url, 35
  ldap-server, 35
  limits, 34
  listener, 33
  log-events, 34
  logging, 34
  mac, 39
  ocsp-responder, 36
  params, 31
  private, 32
  public, 33
  radius-server, 44
  radius-shared-secret, 44
  rekey, 38
  rule, 47
  selector, 38, 41
  selector/blackboard, 42
  selector/certificate, 41
  selector/host-certificate, 42
  selector/interface, 38, 42

selector/ip, 38, 42
selector/publickey-passed, 43
selector/user, 42
selector/user-group, 42
selector/user-password-change-needed, 43
selector/user-privileged, 42
services, 46
settings, 31
socks-server-url, 35
submethod-generic, 44
submethod-pam, 44
submethod-password, 44
submethod-radius, 44
submethod-securid, 44
subsystem, 48
terminal, 48
tunnel-agent, 49
tunnel-local, 49
tunnel-local/dst, 50
tunnel-local/src, 50
tunnel-remote, 50
tunnel-remote/listen, 50
tunnel-remote/src, 50
tunnel-x11, 49
x509-certificate, 33